
Abjad Documentation

Release 2.10

Trevor Bača, Josiah Oberholtzer, Víctor Adán

October 05, 2012

CONTENTS

1	Start here	3
1.1	Abjad?	3
1.2	Installation	4
1.3	Version history	6
2	Examples	43
2.1	Bartók: <i>Mikrokosmos</i>	43
2.2	Ferneyhough: <i>Unsichtbare Farben</i>	47
2.3	Ligeti: <i>Désordre</i>	50
2.4	Mozart: <i>Musikalisches Würfelspiel</i>	54
3	System Overview	67
3.1	Leaf, Container, Spanner, Mark	67
3.2	Parsing	70
4	Tutorials	77
4.1	Changing notes to rests	77
4.2	Creating rest-delimited slurs	78
4.3	Making grob overrides	79
4.4	Understanding LilyPond grobs	80
4.5	Understanding time signature marks	82
4.6	Working with component parentage	90
4.7	Working with threads	91
5	Reference manual	111
5.1	Annotations	111
5.2	Articulations	112
5.3	Chords	118
5.4	Containers	122
5.5	Durations	129
5.6	Instrument marks	137
5.7	I/O	139
5.8	LilyPond command marks	140
5.9	LilyPond comments	143
5.10	LilyPond files	145
5.11	Measures	147
5.12	Notes	150
5.13	Pitches	153
5.14	Working with lists of numbers	159

5.15	Rests	160
5.16	Scores	161
5.17	Spanners	164
5.18	Staves	166
5.19	Tuplets	169
5.20	Voices	174
6	Developer documentation	179
6.1	Codebase	179
6.2	Docs	181
6.3	Tests	186
6.4	Scripts	188
6.5	Using <code>abjad-book</code>	190
6.6	Timing code	193
6.7	Profiling code	193
6.8	Memory consumption	194
6.9	Class attributes	194
6.10	Using slots	196
6.11	Coding standards	197
7	Appendices	201
7.1	From Trevor and Víctor	201
7.2	Why MIDI is not enough	202
7.3	Why LilyPond is right for Abjad	204
7.4	LilyPond text alignment	206
7.5	Score Snippet Gallery	207
7.6	Change log	208
7.7	Bibliography	221
8	Abjad API	223
8.1	Abjad API	223
	Bibliography	2107
	Index	2109

Abjad helps composers build up complex pieces of music notation in an iterative and incremental way. Use Abjad to create a symbolic representation of all the notes, rests, staves, tuplets, beams and slurs in any score. Because Abjad extends the Python programming language, you can use Abjad to make systematic changes to your music as you work. And because Abjad wraps the powerful LilyPond music notation package, you can use Abjad to control the typographic details of the symbols on the page.



START HERE

1.1 Abjad?

Abjad is an interactive software system designed to help composers build up complex pieces of music notation in an iterative and incremental way. Use Abjad to create a symbolic representation of all the notes, rests, staves, tuplets, beams and slurs in any score. Because Abjad extends the Python programming language, you can use Abjad to make systematic changes to your music as you work. And because Abjad wraps the powerful LilyPond music notation package, you can use Abjad to control the typographic details of the symbols on the page.

1.1.1 Abjad extends LilyPond

[LilyPond](#) is an open-source music notation package invented by Han-Wen Nienhuys and Jan Niewenhuizen and extended by an international team of developers and musicians. LilyPond differs from other music engraving programs in a number of ways. LilyPond separates musical content from page layout. LilyPond affords typographic control over almost everything. And LilyPond implements a powerfully correct model of the musical score.

You can start working with Abjad right away because Abjad creates LilyPond files for you automatically. But you will work with Abjad faster and more effectively if you understand the structure of the LilyPond files Abjad creates. For this reason we recommend new users spend a couple of days learning LilyPond first.

Start by reading about [text input](#) in LilyPond. Then work the [LilyPond tutorial](#). You can test your understanding of LilyPond by using the program to engrave of a Bach chorale. Use a grand staff and include slurs, fermatas and so on. Once you can engrave a chorale in LilyPond you'll understand the way Abjad works with LilyPond behind the scenes.

1.1.2 Abjad extends Python

[Python](#) is an open-source programming language invented by Guido van Rossum and further developed by a team of programmers working in many countries around the world. Python is used to provision servers, process text, develop distributed systems and do much more besides. The dynamic language and interpreter features of Python are similar to Ruby while the syntax of Python resembles C, C++ and Java.

To get the most out of Abjad you need to know (or learn) the basics of programming in Python. Abjad extends Python because it makes no sense to reinvent the wheel modern programming languages have developed to find, sort, store, model and encapsulate information. Abjad simply piggy-backs on the ways of doing these things that Python provides. So to use Abjad effectively you need to know the way these things are done in Python.

Start with the [Python tutorial](#). The tutorial is structured in 15 chapters and you should work through the first 12. This will take a day or two and you'll be able to use all the information you read in the Python tutorial in Abjad. If you're an experienced programmer you should skip chapters 1 - 3 but read 4 - 12. When you're done you can give yourself

the equivalent of the chorale test suggested above. First open a file and define a couple of classes and functions in it. Then open a second file and write some code to first import and then do stuff with the classes and functions you defined in the first file. Once you can easily do this without looking at the Python docs you'll be in a much better position to work with Abjad.

1.1.3 What next?

The most important parts of Abjad are the interlocking objects that structure the system. Read about the way Abjad models pitch, duration, leaves, containers, spanners and marks in the *Abjad reference manual*.

But note that important parts of the system are missing from the manual. The reason for this is that we completed the Abjad API months before we started the manual. This means that classes and functions you look up in the API may not yet be documented in the manual. The reference manual will eventually document all parts of the system. But until then check the API if the manual doesn't yet have what you need.

Once you understand the basics about how to work with Abjad you should spend some time with the *Abjad API*. The API documents all the functionality available in the system. Abjad comprises about 168,000 lines of code. About half of these implement the automated tests that check the correctness of Abjad. The rest of the code implements 39 packages comprising 221 classes and 1029 functions. All of these are documented in the API.

1.1.4 Mailing lists

As you begin working with Abjad please be in touch.

Questions, comments and contributions are welcomed from composers everywhere.

Questions or comments? Join the [abjad-user](#) list.

Want to contribute? Join the [abjad-devel](#) list.

1.2 Installation

1.2.1 Abjad depends on Python

You must have Python 2.7 installed to run Abjad.

Abjad does not yet support the Python 3.x series of releases.

To check the version of Python installed on your computer type the following:

```
python --version
```

You can download different versions of Python at <http://www.python.org>.

1.2.2 Abjad depends on LilyPond

You must have LilyPond 2.16 or greater installed for Abjad to work properly.

You can download LilyPond at <http://www.lilypond.org>.

After you have installed LilyPond you should type the following to see if LilyPond is callable from your commandline:

```
lilypond --version
```

If LilyPond is not callable from your commandline you should add the location of the LilyPond executable to your `PATH` environment variable.

If you are new to working with the commandline you should use Google to get a basic introduction to editing your profile and setting environment variables.

1.2.3 Installing the current packaged version of Abjad with `easy_install`

There are different ways to install Python packages on your computer. One of the most direct ways is with `easy_install`.

If you have `easy_install` installed on your computer then you can install Abjad with this command:

```
sudo easy_install -U abjad
```

Python will install Abjad in the site packages directory on your computer and you'll be ready to start using the system.

If you do not have `easy_install` installed on your computer then you should follow the instructions below to install the current packaged version of Abjad from the Python Package Index.

1.2.4 Installing the current packaged version of Abjad from the Python Package Index

If you do not have `easy_install` installed on your computer you should follow these steps to install the current packaged version of Abjad from the Python Package Index:

1. Download the current release of Abjad from <http://pypi.python.org/pypi/Abjad>.
2. Unarchive the downloaded file. Under MacOS and Windows you can double click the archived file.

Under Linux execute the following command with `x.y` replaced by the current release of Abjad:

```
tar xzvf Abjad-x.y.tar.gz
```

3. Change into the directory created in step 2:

```
cd Abjad-x.y
```

4. Run the following under MacOS or Linux:

```
sudo python setup.py install
```

5. Or run this command under Windows after starting up a command shell with administrator privileges:

```
setup.py install
```

These commands will cause Python to install Abjad in your site packages directory. You'll then be ready to start using Abjad.

1.2.5 After install

When first run, Abjad creates an `.abjad` directory in your own `$HOME` directory. In `$HOME/.abjad` you will find the Abjad configuration file: `config.py`. Here you can tell Abjad about your preferred PDF file viewer, MIDI player, your preferred LilyPond language, etc. All relevant variables have defaults that you can change to suit your needs. In Linux, for example, you might want to set your `pdfviewer` to `evince` and your `MIDIplayer` to `tiMIDiTy`.

`config.py` is a regular Python file, so you should make sure the file follows Python syntax.

1.2.6 Note for Linux users

Abjad defaults to `xdg-open` to display PDF files using your default PDF viewer. Most Linux distributions now come with `xdg-utils` installed.

If you do not have `xdg-utils` installed, you can download it from <http://www.portland.freedsektop.org>.

Alternatively you can set the `pdfviewer` variable in `$HOME/.abjad/config` to your favorite PDF viewer.

1.3 Version history

1.3.1 Abjad 2.10

Released 2012-10-05. Built from r7615. Implements 437 public classes and 982 functions totalling 179,000 lines of code.

The following packages now load by default when you start Abjad:

```
Abjad 2.10
>>> [x for x in dir() if x.endswith('tools')]
['abjadbooktools', 'beamtools', 'chordtools', 'componenttools', 'containertools', 'contexttools',
'developerscripttools', 'durationtools', 'formattools', 'gracetools', 'instrumenttools',
'introspectiontools', 'iotools', 'iterationtools', 'labeltools', 'layouttools', 'leaftools',
'lilypondfiletools', 'marktools', 'markuptools', 'mathtools', 'measuretools', 'notetools',
'offsettools', 'pitcharraytools', 'pitchtools', 'resttools', 'rhythmtreertools', 'schemetools',
'scoretemplatetools', 'scoretools', 'sequencetools', 'sievetools', 'skiptools', 'spannertools',
'stafftools', 'stringtools', 'tempotools', 'tietools', 'timeintervaltools', 'timesignaturetools',
'timetokentools', 'tonalitytools', 'tupletools', 'verticalitytools', 'voicetools']
```

Improved formatting engine. Scores now format approximately 30% faster.

Improved LilyPond parser.

Markup objects now parse input string input on initialization:

```
>>> markuptools.Markup(r'\bold \tiny { foo bar baz }')
Markup((MarkupCommand('bold', MarkupCommand('tiny', ['foo', 'bar', 'baz']))),))

>>> print _.indented_lilypond_format
\markup {
  \bold
    \tiny
      {
        foo
        bar
        baz
      }
}
```

You can now context names to reference named contexts attached to any container:

```
>>> template = scoretemplatetools.StringQuartetScoreTemplate()
>>> score = template()

>>> score['First Violin Staff']
Staff-"First Violin Staff"{1}
```

```
>>> score['First Violin Voice']
Voice-"First Violin Voice"{}

```

Five new constants are available globally.

- The constants are Left, Right, Up, Down and Center.
- The constants function like Python's built-in True and False.
- Use the constants as keyword defaults.

A new configuration tool is available:

```
configurationtools.get_abjad_startup_string()
```

New context tools are available:

```
contexttools.all_are_contexts()
```

A new iterationtools package is available:

```
iterationtools.iterate_chords_in_expr()
iterationtools.iterate_components_and_grace_containers_in_expr()
iterationtools.iterate_components_depth_first()
iterationtools.iterate_components_in_expr()
iterationtools.iterate_containers_in_expr()
iterationtools.iterate_contexts_in_expr()
iterationtools.iterate_leaf_pairs_in_expr()
iterationtools.iterate_leaves_in_expr()
iterationtools.iterate_measures_in_expr()
iterationtools.iterate_namesakes_from_component()
iterationtools.iterate_notes_and_chords_in_expr()
iterationtools.iterate_notes_in_expr()
iterationtools.iterate_rests_in_expr()
iterationtools.iterate_scores_in_expr()
iterationtools.iterate_semantic_voices_in_expr()
iterationtools.iterate_skips_in_expr()
iterationtools.iterate_staves_in_expr()
iterationtools.iterate_thread_from_component()
iterationtools.iterate_thread_in_expr()
iterationtools.iterate_timeline_from_component()
iterationtools.iterate_timeline_in_expr()
iterationtools.iterate_tuplets_in_expr()
iterationtools.iterate_voices_in_expr()

```

New LilyPond file tools are available:

```
lilypondfiletools.make_floating_time_signature_lilypond_file()
```

New LilyPond parser tools are available:

```
lilypondparsertools.GuileProxy
lilypondparsertools.LilyPondDuration
lilypondparsertools.LilyPondEvent
lilypondparsertools.LilyPondFraction
lilypondparsertools.LilyPondLexicalDefinition
lilypondparsertools.LilyPondSyntacticalDefinition
lilypondparsertools.ReducedLyParser
lilypondparsertools.SchemeParser
lilypondparsertools.SyntaxNode
lilypondparsertools.lilypond_enharmonic_transpose()

```

A new `Ratio` class is available in the `mathtools` package:

```
>>> mathtools.Ratio(1, 2, -1)
Ratio(1, 2, -1)
```

New rhythm-tree tools are available.

- Implemented RTM expression parser:

```
rhythmtreetools.RhythmTreeParser
```

- Implemented new classes for explicitly constructing rhythm-trees:

```
RhythmTreeNode
RhythmTreeLeaf
RhythmTreeContainer
```

```
>>> from abjad import *
>>> rtm = '(1 (1 (2 (1 -1 1)) -2))'
>>> result = rhythmtreetools.RhythmTreeParser()(rtm)

>>> result[0]
RhythmTreeContainer(
  children=(
    RhythmTreeLeaf(
      duration=1,
      pitched=True,
    ),
    RhythmTreeContainer(
      children=(
        RhythmTreeLeaf(
          duration=1,
          pitched=True,
        ),
        RhythmTreeLeaf(
          duration=1,
          pitched=False,
        ),
        RhythmTreeLeaf(
          duration=1,
          pitched=True,
        ),
      ),
      duration=2
    ),
    RhythmTreeLeaf(
      duration=2,
      pitched=False,
    ),
  ),
  duration=1
)

>>> _._rtm_format
'(1 (1 (2 (1 -1 1)) -2))'

>>> result[0]((1, 4))
FixedDurationTuplet(1/4, [c'16, {@ 3:2 c'16, r16, c'16 @}, r8])
```



```
>>> f(_)
\times 4/5 {
    c'16
    \times 2/3 {
        c'16
        r16
        c'16
    }
    r8
}
```

New Scheme tools are available.

- Added `force_quotes` boolean keyword to `schemetools.Scheme` and `schemetools.format_scheme_value()`:

```
>>> schemetools.format_scheme_value('foo')
'foo'

>>> schemetools.format_scheme_value('foo', force_quotes=True)
'"foo"'
```

This allows you to force double quotes around strings which contain no spaces. This is necessary for some LilyPond grob overrides.

- A new Scheme formatting function is available:

```
schemetools.format_scheme_value()
```

New score-template tools are available:

```
scoretemplatetools.GroupedStavesScoreTemplate
```

New sequence tools are available:

- Added `sequencetools.merge_duration_sequences()`:

```
>>> sequencetools.merge_duration_sequences([10, 10, 10], [7])
[7, 3, 10, 10]
```
- Added `sequencetools.pair_duration_sequence_elements_with_input_pair_values()`:

```
>>> duration_sequence = [10, 10, 10, 10]
>>> input_pairs = [('red', 1), ('orange', 18), ('yellow', 200)]
>>> sequencetools.pair_duration_sequence_elements_with_input_pair_values(
... duration_sequence, input_pairs)
[(10, 'red'), (10, 'orange'), (10, 'yellow'), (10, 'yellow')]
```

New tie tools are available:

```
tietools.get_tie_spanner_attached_to_component()
```

New time-interval tools are available:

```
timeintervaltools.make_voice_from_nonoverlapping_intervals()
```

New time-token tools are available:

- Added `SkipFilledTimeTokenMaker` to `timetokentools` package:

```
>>> maker = timetokentools.SkipFilledTimeTokenMaker()
```

```
>>> duration_tokens = [(1, 5), (1, 4), (1, 6), (7, 9)]
>>> leaf_lists = maker(duration_tokens)
>>> leaves = sequencetools.flatten_sequence(leaf_lists)

>>> staff = Staff(leaves)

>>> f(staff)
\new Staff {
  s1 * 1/5
  s1 * 1/4
  s1 * 1/6
  s1 * 7/9
}
```

- Added `TupletMonadTimeTokenMaker` to `timetoken tools` package:

```
>>> maker = timetoken tools.TupletMonadTimeTokenMaker()

>>> duration_tokens = [(1, 5), (1, 4), (1, 6), (7, 9)]
>>> tuplets = maker(duration_tokens)
>>> staff = Staff(tuplets)

>>> f(staff)
\new Staff {
  \times 4/5 {
    c'4
  }
  {
    c'4
  }
  \times 2/3 {
    c'4
  }
  \times 8/9 {
    c'2..
  }
}
```

1.3.2 Abjad 2.9

Released 2012-06-05. Built from r5795. Implements 405 public classes and 1066 functions totalling 182,000 lines of code.

Extended markup handling is now available.

- The LilyPond parser accepts complex markup as input:

```
>>> f(p(r'''{ c'4 _ \markup { \put-adjacent #1 #-1 \bold \fontsize #2 \upright foo bar } }'''))
{
  c'4
  _ \markup {
    \put-adjacent
      #1
      #-1
      \bold
      \fontsize
        #2
        \upright
  }
}
```

```

        bar
    }
}

```

- Format routines allow for markup indentation:

```

>>> circle = markuptools.MarkupCommand('draw-circle', 2.5, 0.1, False)
>>> square = markuptools.MarkupCommand('rounded-box', 'hello?')
>>> line = markuptools.MarkupCommand('line', [square, 'wow!'])
>>> markup = markuptools.Markup(('X', square, 'Y', line, 'Z'), direction='up')

>>> print '\n'.join(markup._get_format_pieces(is_indented=True))
^ \markup {
  X
  \rounded-box
    hello?
  Y
  \line
    {
      \rounded-box
        hello?
      wow!
    }
  Z
}

```

- Nontrivial markup format with indentation automatically:

```

>>> staff = Staff("c")
>>> m1 = markuptools.Markup('foo') (staff[0])
>>> m2 = markuptools.Markup('bar') (staff[0])
>>> m3 = markuptools.Markup('baz', 'up') (staff[0])
>>> m4 = markuptools.Markup('quux', 'down') (staff[0])
>>> accent = marktools.Articulation('accent') (staff[0])

>>> f(staff)
\new Staff {
  c4 -\accent
    ^ \markup { baz }
    _ \markup { quux }
    - \markup {
      \column
      {
        foo
        bar
      }
    }
}

```

- Markup.contents is now a tuple of strings or MarkupCommand instances.
- Removed the markup style_string property. Use schemetools classes for constructing Scheme-style formatting.
- Changed Markup.contents_string to Markup.contents.

An entirely new tuplet microlanguage is now available.

- This “reduced ly” syntax uses braces to show tuplet nesting and represents rhythm without pitch:

```
>>> from abjad.tools import rhythmtreetools

>>> container = rhythmtreetools.parse_reduced_ly_syntax('4 -4 8 5/3 { 2/3 { 8 8 8 } { 8 8 } -8 }

>>> f(container)
{
  c'4
  r4
  c'8
  \fraction \times 5/3 {
    \times 2/3 {
      c'8
      c'8
      c'8
    }
    {
      c'8
      c'8
    }
    r8
  }
  c'4
}
```

- Measures and dotted values are also available:

```
>>> container = rhythmtreetools.parse_reduced_ly_syntax('|2/4 8. 16 8. 16| |4/4 2/3 { 2 2 2 }|')

f(container)
{
  {
    \time 2/4
    c'8.
    c'16
    c'8.
    c'16
  }
  {
    \time 4/4
    \times 2/3 {
      c'2
      c'2
      c'2
    }
  }
}
```

Extended container input syntax.

- You can now pass strings directly to the `append()` and `extend()` methods of any container:

```
>>> container = Container()
>>> container
{}

>>> container.extend('a b c')
>>> container
{a4, b4, c4}
```

```
>>> container.append('d')
>>> container
{a'4, b'4, c'4, d'4}
```

- You can assign a string to any container item:

```
>>> container = Container("c' d' e'")
>>> container
{c'4, d'4, e'4}

>>> container[1] = 'r'
>>> container
{c'4, r4, e'4}
```

- You can assign a string to any container slice:

```
>>> container = Container("c' d' e'")
>>> container
{c'4, d'4, e'4}

>>> container[:2] = 'r8 r r'
>>> container
{r8, r8, r8, e'4}
```

- You can initialize containers from strings using alternate parsers.

Use the 'abj' prefix to initialize a container with the new reduced ly syntax:

```
>>> staff = Staff('abj: | 2/4 2/3 { 8 4 } 8 8 || 3/4 4 4 4 |')

>>> f(staff)
\new Staff {
  {
    \time 2/4
    \times 2/3 {
      c'8
      c'4
    }
    c'8
    c'8
  }
  {
    \time 3/4
    c'4
    c'4
    c'4
  }
}
```

- Use the 'rtm' prefix to initialize a container with IRCAM RTM-style syntax:

```
>>> staff = Staff('rtm: (1 (1 (2 (1 1 1)) 1)) (1 (1 1))')

>>> f(staff)
\new Staff {
  c'16
  \times 2/3 {
    c'16
    c'16
    c'16
  }
}
```

```

    }
    c'16
    c'8
    c'8
}

```

- Parallel contexts, such as `Score`, can be instantiated from strings which parse to a sequence of contexts:

```
Score(r'''\new Staff { c' } \new Staff = { c, }''')
```

- Added a new `FixedDurationContainer` class to the `containertools` package.

Fixed-duration containers extend container behavior with format-time checking against a user-specified target duration:

```

>>> container = containertools.FixedDurationContainer((3, 8), "c'8 d'8 e'8")

>>> container
FixedDurationContainer(Duration(3, 8), [Note("c'8"), Note("d'8"), Note("e'8")])

>>> f(container)
{
    c'8
    d'8
    e'8
}

>>> container.is_misfilled
False

>>> container.pop()
Note("e'8")

>>> container
FixedDurationContainer(Duration(3, 8), [Note("c'8"), Note("d'8")])

>>> container.is_misfilled
True

```

Misfilled fixed-duration containers will raise an exception at format-time. Fixed-duration containers share this behavior with measures.

Regularized measure modification behavior.

- By default measures do not automatically adjust time signature after contents modification:

```

>>> measure = Measure((3, 4), "c' d' e'")
>>> measure
Measure(3/4, [c'4, d'4, e'4])

>>> measure.append('r')
>>> measure
Measure(3/4, [c'4, d'4, e'4, r4])

>>> measure.is_overfull
True

```

- But it is now possible to cause measures to automatically adjust time signature after contents modification:

```
>>> measure = Measure((3, 4), "c' d' e'")
>>> measure.automatically_adjust_time_signature = True
>>> measure
Measure(3/4, [c'4, d'4, e'4])

>>> measure.append('r')
>>> measure
Measure(4/4, [c'4, d'4, e'4, r4])

>>> measure.is_misfilled
False
```

Previous implementations of `measure.append()`, `extend()` and `set-item` never adjusted measure time signatures.

Now the behavior of such operations is controllable on a measure-by-measure basis by the end user.

New functionality is available for working with ties.

- Added a `TieChain` class to the `tietools` package. Tie chains now return as a custom `TieChain` object instead of tuple:

```
>>> staff = Staff("c' d' e' ~ e'")

>>> tietools.get_tie_chain(staff[2])
TieChain((Note("e'4"), Note("e'4")))
```

Reimplemented tie chain duration attributes as explicit class attributes. The following four functions have been removed:

```
tietools.get_preprolated_tie_chain_duration()
tietools.get_prolated_tie_chain_duration()
tietools.get_tie_chain_duration_in_seconds()
tietools.get_written_tie_chain_duration()
```

Use these read-only properties instead:

```
TieChain.preprolated_duration
TieChain.prolated_duration
TieChain.duration_in_seconds
TieChain.written_duration
```

The `TieChain` class inherits from the new `ScoreSelection` abstract base class.

Added new `tietools` functions:

```
tietools.iterate_pitched_tie_chains_forward_in_expr()
tietools.iterate_pitched_tie_chains_backward_in_expr()
tietools.iterate_nontrivial_tie_chains_forward_in_expr()
tietools.iterate_nontrivial_tie_chains_backward_in_expr()
```

Removed `tietools.is_tie_chain(expr)`. Use `isinstance(expr, tietools.TieChain)` instead.

Removed `tietools.get_leaves_in_tie_chain()`. Use `TieChain.leaves` instead.

Removed `tietools.group_leaves_in_tie_chain_by_immediate_parents()`. Use `TieChain.leaves_grouped_by_immediate_parents` instead.

Removed `tietools.is_tie_chain_with_all_leaves_in_same_parent()`. Use `TieChain.all_leaves_are_in_same_parent` instead.

Added a new `stringtools` package.

- The following functions all migrated from the `iotools` package:

```
stringtools.capitalize_string_start()
stringtools.format_input_lines_as_doc_string()
stringtools.format_input_lines_as_regression_test()
stringtools.is_lowercamelcase_string()
stringtools.is_space_delimited_lowercase_string()
stringtools.is_underscore_delimited_lowercase_file_name()
stringtools.is_underscore_delimited_lowercase_file_name_with_extension()
stringtools.is_underscore_delimited_lowercase_package_name()
stringtools.is_underscore_delimited_lowercase_string()
stringtools.is_uppercamelcase_string()
stringtools.space_delimited_lowercase_to_uppercamelcase()
stringtools.string_to_strict_directory_name()
stringtools.strip_diacritics_from_binary_string()
stringtools.underscore_delimited_lowercase_to_lowercamelcase()
stringtools.underscore_delimited_lowercase_to_uppercamelcase()
stringtools.uppercamelcase_to_space_delimited_lowercase()
stringtools.uppercamelcase_to_underscore_delimited_lowercase()
```

The package also contains these new functions:

```
stringtools.arg_to_bidirectional_direction_string()
stringtools.arg_to_bidirectional_lilypond_symbol()
stringtools.arg_to_tridirectional_direction_string()
stringtools.arg_to_tridirectional_lilypond_symbol()

>>> stringtools.arg_to_bidirectional_lilypond_symbol(1)
'^'
>>> stringtools.arg_to_tridirectional_direction_string('-')
'neutral'
```

Added a new `beamtools` package.

- This release of the `beamtools` package contains the following classes and functions:

```
beamtools.BeamSpanner
beamtools.ComplexBeamSpanner
beamtools.DuratedComplexBeamSpanner
beamtools.MultipartBeamSpanner

beamtools.is_beamable_component
beamtools.apply_beam_spanner_to_measure
beamtools.apply_beam_spanners_to_measures_in_expr
beamtools.apply_complex_beam_spanner_to_measure
beamtools.apply_complex_beam_spanners_to_measures_in_expr
beamtools.apply_durated_complex_beam_spanner_to_measures
beamtools.beam_bottommost_tuplets_in_expr
beamtools.get_beam_spanner_attached_to_component
beamtools.is_beamable_component
beamtools.is_component_with_beam_spanner_attached
```

Note that the following two functions have been removed:

```
beamtools.apply_beam_spanner_to_measure()
beamtools.apply_complex_beam_spanner_to_measure()
```

Use these two functions instead:


```
beamtools.apply_beam_spanners_to_measures_in_expr()
beamtools.apply_complex_beam_spanners_to_measures_in_expr()
```

New `constrainttools` functionality is now available.

- Extended the `VariableLengthStreamSolver` class.

The class now produces more randomly ordered solution sets than before, when in randomized mode. Note that the solution sets tend to increase in size. Also note that there is an increased performance hit for such PMC-style randomized constraint solving:

```
>>> from abjad.tools.constrainttools import *

>>> domain = Domain([1, 2, 3, 4], 1)
>>> boundary_sum = GlobalConstraint(lambda x: sum(x) < 6)
>>> target_sum = GlobalConstraint(lambda x: sum(x) == 5)
>>> random_solver = VariableLengthStreamSolver(domain,
... [boundary_sum], [target_sum], randomized=True)
>>> for x in random_solver: x
...
[1, 3, 1]
[4, 1]
[3, 2]
[2, 3]
[1, 4]
[3, 1, 1]
[2, 1, 2]
[1, 2, 1, 1]
[2, 1, 1, 1]
[2, 2, 1]
[1, 1, 1, 2]
[1, 2, 2]
[1, 1, 1, 1, 1]
[1, 1, 3]
[1, 1, 2, 1]
```

- Randomized the `FixedLengthStreamSolvers` class.

The class now produces truly randomly ordered solution sets.

New sequence tools are available.

- Added new type- and form-checking predicates to the `sequencetools` package:

```
sequencetools.all_are_integer_equivalent_exprs
sequencetools.is_null_tuple(expr)
sequencetools.is_singleton(expr)
sequencetools.is_pair(expr)
sequencetools.is_n_tuple(expr, n)
sequencetools.is_integer_singleton(expr)
sequencetools.is_integer_pair(expr)
sequencetools.is_integer_n_tuple(expr, n)
sequencetools.is_integer_equivalent_n_tuple
sequencetools.is_integer_equivalent_pair
sequencetools.is_integer_equivalent_singleton
sequencetools.is_fraction_equivalent_pair
```

Each function returns a boolean:

```
>>> sequencetools.is_integer_singleton((19,))
True
```

- Added a new `NonreducedFraction` class to the `sequencetools` package:

```
>>> sequencetools.NonreducedFraction(3, 6)
NonreducedFraction(3, 6)
```

Like built-in fraction but numerator and denominator do NOT simplify.

All six comparators are implemented on nonreduced fractions.

Addition and subtraction are implemented on nonreduced fractions:

```
>>> sequencetools.NonreducedFraction(3, 6) + sequencetools.NonreducedFraction(3, 6)
NonreducedFraction(6, 6)
```

Use nonreduced fractions to model arithmetic operations on time signature-like objects absent any of the special time signature features like partial-measure pick-ups.

New spanners and spanner handlers are now available.

- Added a `ComplexGlissandoSpanner` to the `spannertools` package.

This spanner generates a glissando which skips over rests. It can be used in combination with `spannertools.BeamSpanner` and an override of the Stem grob to generate the appearance of durated glissandi:

```
>>> staff = Staff("c'16 [ d' r e' r r r g' ]")

>>> f(staff)
\new Staff {
  c'16 [
    d'16
    r16
    e'16
    r16
    r16
    r16
    g'16 ]
}

>>> spannertools.ComplexGlissandoSpanner(staff[:])
ComplexGlissandoSpanner(c'16, d'16, r16, e'16, r16, r16, r16, g'16)

>>> staff.override.stem.stemlet_length = 2
>>> f(staff)
\new Staff \with {
  \override Stem #'stemlet-length = #2
} {
  c'16 [ \glissando
  d'16 \glissando
  \once \override NoteColumn #'glissando-skip = ##t
  \once \override Rest #'transparent = ##t
  r16
  e'16 \glissando
  \once \override NoteColumn #'glissando-skip = ##t
  \once \override Rest #'transparent = ##t
  r16
  \once \override NoteColumn #'glissando-skip = ##t
  \once \override Rest #'transparent = ##t
  r16
}
```

```
\once \override NoteColumn #'glissando-skip = ##t
\once \override Rest #'transparent = ##t
r16
g'16 ]
}
```

- Added new `spannertools` function:

```
spannertools.destory_spanners_attached_to_components_in_expr(expr, klass=None)
```

The function can be useful for removing all spanners when debugging a complex expression.

- Spanners are now callable:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = spannertools.BeamSpanner()
>>> beam(staff[:])
Staff{4}

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

This works the same way as marks:

```
>>> marktools.Articulation('.') (staff[1])
Articulation('.') (d'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8 -\staccato
  e'8
  f'8 ]
}
```

Callable spanners are provided as an experimental way of unifying the attachment syntax of spanners and marks.

Many new functions are available in the `componenttools` package.

- New getters:

```
componenttools.get_proper_contents_of_component()
componenttools.get_improper_contents_of_component()
componenttools.get_improper_contents_of_component_that_start_with_component()
componenttools.get_improper_contents_of_component_that_stop_with_component()
componenttools.get_proper_descendents_of_component()
componenttools.get_improper_descendents_of_component()
componenttools.get_improper_descendents_of_component_that_cross_prolated_offset
componenttools.get_improper_descendents_of_component_that_start_with_component
componenttools.get_improper_descendents_of_component_that_stop_with_component
componenttools.get_lineage_of_component()
componenttools.get_lineage_of_component_that_start_with_component()
componenttools.get_lineage_of_component_that_stop_with_component()
componenttools.get_nth_sibling_from_component(component, n)
componenttools.get_nth_component_from_component_in_time_order(component, n)
```

```
componenttools.get_nth_namesake_from_component  
componenttools.get_most_distant_sequential_container_in_improper_parentage_of_component()
```

Use these functions to interrogate the structural relations of components resident inside arbitrarily complex pieces of score.

The functions are useful as primitive methods when implementing more complex operations designed to mutate the score tree.

- Note the difference between the ‘contents’ of a component and the ‘descendents’ of a component:

```
>>> componenttools.get_proper_contents_of_component(staff)  
[Note("c'4"), Tuplet(2/3, [d'8, e'8, f'8])]
```

Versus:

```
>>> componenttools.get_proper_descendents_of_component(staff)  
[Note("c'4"), Tuplet(2/3, [d'8, e'8, f'8]), Note("d'8"), Note("e'8"), Note("f'8")]
```

- Also add the following `componenttools` predicate:

```
componenttools.is_immediate_temporal_successor_of_component()
```

Further new functionality:

- Added new `gracetools` function:

```
gracetools.detach_grace_containers_attached_to_leaves_in_expr()
```

Use the function to strip all grace containers from an arbitrary piece of score.

- Added new `marktools` functions:

```
marktools.get_marks_attached_to_components_in_expr()  
marktools.detach_marks_attached_to_components_in_expr()  
marktools.move_marks(donor, recipient).
```

- Added new `pitchtools` function:

```
pitchtools.set_written_pitch_of_pitched_components_in_expr(expr, written_pitch=0)
```

Use the function to neutralize pitch information in an arbitrary piece of score.

- Added new `tuplettools` functions:

```
tuplettools.change_fixed_duration_tuplets_in_expr_to_tuplets()  
tuplettools.change_tuplets_in_expr_to_fixed_duration_tuplets()
```

- Extended `lilypondfiletools.ContextBlock` with the following attributes:

```
ContextBlock.engraver_consists  
ContextBlock.engraver_removals  
ContextBlock.context_name  
ContextBlock.name  
ContextBlock.type
```

The attributes correspond to backslash-initiated LilyPond commands available in LilyPond context blocks.

- Updated `LilyPondLanguageToken` to format LilyPond `\language` command instead of LilyPond `\include` command.
- Extended `Duration` to initialize from LilyPond duration strings:

```
>>> Duration('8.')
Duration(3, 16)
```

Note that this means that `Duration('2')` now gives `Duration(1, 2)`. Previously `Duration('2')` gave `Duration(2, 1)` just like `Fraction('2')`.

Changes to end-user functionality:

- Changed:

```
componenttools.copy_components_and_remove_all_spanners()

componenttools.copy_components_and_remove_spanners()
```

- Changed:

```
componenttools.get_improper_contents_of_component_that_cross_prolated_offset()

componenttools.get_leftmost_components_with_prolated_duration_at_most()
```

- Changed:

```
componenttools.list_improper_contents_of_component_that_cross_prolated_offset()

componenttools.list_leftmost_components_with_prolated_duration_at_most()
```

- Changed:

```
configurationtool.set_default_accidental_spelling()

pitchtools.set_default_accidental_spelling()
```

- Changed:

```
gracetools.Grace

gracetools.GraceContainer
```

- Changed:

```
spannertools.destory_all_spanners_attached_to_component()

spannertools.destory_spanners_attached_to_component()
```

- Changed:

```
spannertools.fracture_all_spanners_attached_to_component()

spannertools.fracture_spanners_attached_to_component()
```

- Changed:

```
spannertools.report_as_string_format_contributions_of_all_spanners_attached_to_component()

spannertools.report_as_string_format_contributions_of_spanners_attached_to_component()
```

- Changed:

```
spannertools.report_as_string_format_contributions_of_all_spanners_attached_to_improper_parentag
```

```
spannertools.report_as_string_format_contributions_of_spanners_attached_to_improper_parentage_of
```

- Changed:

```
tietools.get_tie_chains_in_expr()
```

```
tietools.get_nontrivial_tie_chains_masked_by_components()
```

- Changed:

```
tietools.remove_all_leaves_in_tie_chain_except_first()
```

```
tietools.remove_nonfirst_leaves_in_tie_chain()
```

- Changed:

```
scr/devel/rename-public-helper
```

```
scr/devel/rename-public-function
```

- Removed the `threadtools` package and moved all functions to `componenttools`.

Instead of these:

```
threadtools.iterate_thread_backward_from_component()
```

```
threadtools.iterate_thread_backward_in_expr()
```

```
threadtools.iterate_thread_forward_from_component()
```

```
threadtools.iterate_thread_forward_in_expr()
```

```
threadtools.component_to_thread_signature()
```

Use these:

```
componenttools.iterate_thread_backward_from_component()
```

```
componenttools.iterate_thread_backward_in_expr()
```

```
componenttools.iterate_thread_forward_from_component()
```

```
componenttools.iterate_thread_forward_in_expr()
```

```
componenttools.component_to_containment_signature()
```

- Removed the read-only `Component.marks` property entirely.
- Removed the top-level `abjad/exceptions` directory. Use the new `exceptiontools` package instead.
- Removed the top-level `abjad/templates` directory.

Make sure to read the changes carefully.

If you have been working with grace notes, for example, you will need to change all occurrences of `gracetools.Grace` to `gracetools.GraceContainer`.

1.3.3 Abjad 2.8

Released 2012-04-16. Built from r5421. Implements 306 public classes and 1037 functions totalling 178,000 lines of code.

Many documentation improvements appear in this release.

- A source link now accompanies all classes and functions in the API:

Source code for abjad.tools.chordtools.arppeggiate_chord

```

from abjad.tools.chordtools.Chord import Chord
from abjad.tools.decoratortools import requires

@requires(Chord)
def arppeggiate_chord(chord):
    '''.. versionadded:: 1.1

    Arpeggiate `chord`:

        abjad> chord = Chord("<c' d' ef'>8")

    ::

        abjad> chordtools.arppeggiate_chord(chord)
        [Note("c'8"), Note("d'8"), Note("ef'8")]

    Arpeggiated notes inherit `chord` written duration.

    Arpeggiated notes do not inherit other `chord` attributes.

    Return list of newly constructed notes.

    .. versionchanged:: 2.0
        renamed ``chordtools.arppeggiate()`` to
        ``chordtools.arppeggiate_chord()``.
    '''
    from abjad.tools.notetools.Note import Note

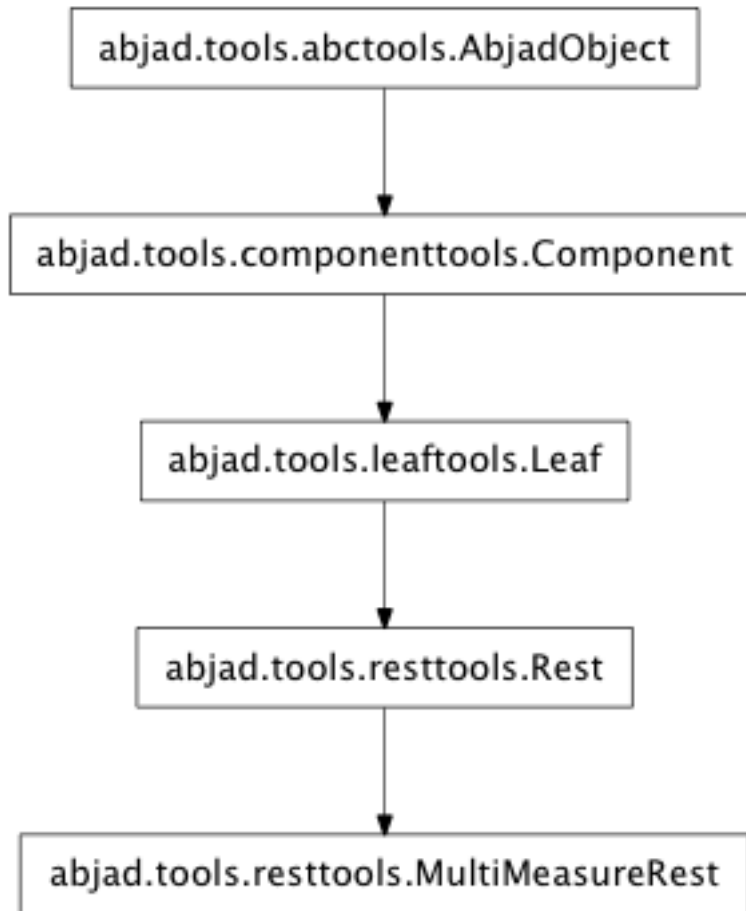
    result = []
    chord_written_duration = chord.written_duration
    for pitch in chord.written_pitches:
        result.append(Note(pitch, chord_written_duration))

    return result

```

[docs]

- All parts of the Abjad codebase are now viewable through the HTML version of the API.
- Inheritance diagrams now accompany all classes:



- Inherited attributes now appear in the API entry of each class.
- Added new `documentationtools` package:

```

documentationtools.APICrawler
documentationtools.AbjadAPIGenerator
documentationtools.ClassCrawler
documentationtools.ClassDocumenter
documentationtools.Documenter
documentationtools.FunctionCrawler
documentationtools.FunctionDocumenter
documentationtools.InheritanceGraph
documentationtools.ModuleCrawler
documentationtools.Pipe
  
```

The package houses custom code to build Abjad documentation.

Added the new `constrainttools` API.

- This release of the `constrainttools` package implements the following classes:

```

constrainttools.AbsoluteIndexConstraint
constrainttools.Domain
constrainttools.FixedLengthStreamSolver
constrainttools.GlobalConstraint
constrainttools.GlobalCountsConstraint
constrainttools.GlobalReferenceConstraint
constrainttools.RelativeCountsConstraint
  
```



```
constrainttools.RelativeIndexConstraint
constrainttools.VariableLengthStreamSolver
```

- Example:

```
>>> from abjad.tools.constrainttools import *

>>> domain = Domain([1, 2, 3, 4], 4)

>>> all_unique = GlobalCountsConstraint(lambda x: all([y == 1 for y in x.values()]))
>>> max_interval = RelativeIndexConstraint([0, 1], lambda x, y: abs(x - y) < 3)
>>> solver = FiniteStreamSolver(domain, [all_unique, max_interval])

>>> for solution in solver: print solution
...
(1, 2, 3, 4)
(1, 2, 4, 3)
(1, 3, 2, 4)
(1, 3, 4, 2)
(2, 1, 3, 4)
(2, 4, 3, 1)
(3, 1, 2, 4)
(3, 4, 2, 1)
(4, 2, 1, 3)
(4, 2, 3, 1)
(4, 3, 1, 2)
(4, 3, 2, 1)
```

- The `constrainttools` package is considered unstable and will be subject to changes in the next releases of Abjad.

Added octave-transposition mapping model.

- This version of the system contains the following classes:

```
pitchtools.OctaveTranspositionMapping
pitchtools.OctaveTranspositionMappingComponent
pitchtools.OctaveTranspositionMappingInventory
```

- Octave-transposition mappings specify a way to maybe pitches from one registral space to another.
- Use octave-transposition mappings as input to `pitchtools.transpose_chromatic_pitch_number_ty_octave_tr`

Many Abjad classes are now implemented as abstract base classes.

- Abstract base classes provide functionality to child subclasses.
- Abstract base classes can not be instantiated directly.
- The Abjad API now lists abstract classes and concrete classes separately.
- See <http://docs.python.org/library/abc.html> for a description of ABCs in Python.

Added the new `abctools` package to house abstract classes that are core to the Abjad object model.

- This version of the package contains the following classes:

```
abctools.AbjadObject
abctools.AttributeEqualityAbjadObject
abctools.ImmutableAbjadObject
abctools.SortableAttributeEqualityAbjadObject
```

- All Abjad classes now inherit from `AbjadObject`.

Added object inventories for several classes.

- This release contains inventories for the following classes:

```
contexttools.ClefMarkInventory
contexttools.TempoMarkInventory
instrumenttools.InstrumentInventory
markuptools.MarkupInventory
pitchtools.OctaveTranspositionMappingInventory
pitchtools.PitchRangeInventory
scoretools.PerformerInventory
```

- Object inventories model ordered collections of system objects.

Add the new `datastructuretools` package.

- This version of the package includes the following classes:

```
datastructuretools.Digraph
datastructuretools.ImmutableDictionary
datastructuretools.ObjectInventory
```

- Use `datastructuretools.Digraph` to detect cycles in any collection of hashable objects:

```
>>> from abjad.tools.datastructuretools import Digraph

>>> edges = [('a', 'b'), ('a', 'c'), ('a', 'f'), ('c', 'd'), ('d', 'e'), ('e', 'c')]
>>> digraph = Digraph(edges)
>>> digraph
Digraph(edges=[('a', 'c'), ('a', 'b'), ('a', 'f'), ('c', 'd'), ('d', 'e'), ('e', 'c')])

>>> digraph.root_nodes
('a',)
>>> digraph.terminal_nodes
('b', 'f')
>>> digraph.cyclic_nodes
('c', 'd', 'e')
>>> digraph.is_cyclic
True
```

- Use `datastructuretools.ObjectInventory` as the base class for an ordered collection of system objects.
- Object inventories inherit from `list` and are mutable.
- Object inventories extend `append()`, `extend()` and `__contains__()` to allow token input.

Added new `wellformednesstools` package.

- This version of the package implements the following classes:

```
wellformednesstools.BeamedQuarterNoteCheck
wellformednesstools.DiscontiguousSpannerCheck
wellformednesstools.DuplicateIdCheck
wellformednesstools.EmptyContainerCheck
wellformednesstools.IntermarkedHairpinCheck
wellformednesstools.MisduratedMeasureCheck
wellformednesstools.MisfilledMeasureCheck
wellformednesstools.MispitchedTieCheck
wellformednesstools.MisrepresentedFlagCheck
wellformednesstools.MissingParentCheck
wellformednesstools.NestedMeasureCheck
```

```
wellformednesstools.OverlappingBeamCheck
wellformednesstools.OverlappingGlissandoCheck
wellformednesstools.OverlappingOctavationCheck
wellformednesstools.ShortHairpinCheck
```

- The classes check different aspects of score well-formedness.
- To call these classes use `componenttools.is_well_formed_component()` or `componenttools.tabulate_well_formedness_violations_in_expr()`.

Added new `decoratortools` package.

- This version of the package contains only the `requires` decorator.
- The `requires` decorator will be used in later versions of Abjad to specify the input and output types of functions explicitly.
- This will help in the construction of function- and class-population tools.

Added new `scoretemplatetools` package.

- This version of the package implements the following classes:

```
scoretemplatetools.StringQuartetScoreTemplate
scoretemplatetools.TwoStaffPianoScoreTemplate
```

- Example:

```
>>> from abjad.tools import scoretemplatetools

>>> template = scoretemplatetools.StringQuartetScoreTemplate()
>>> score = template()

>>> score
Score-"String Quartet Score"<<1>>

>>> f(score)
\context Score = "String Quartet Score" <<
  \context StaffGroup = "String Quartet Staff Group" <<
    \context Staff = "First Violin Staff" {
      \clef "treble"
      \context Voice = "First Violin Voice" {
      }
    }
    \context Staff = "Second Violin Voice" {
      \clef "treble"
    }
    \context Staff = "Viola Staff" {
      \clef "alto"
    }
    \context Staff = "Cello Staff" {
      \clef "bass"
    }
  >>
>>
```

- Class usage follows a two-step initialize-then-call pattern.

Added new `rhythmtreetools` package for parsing IRCAM-like RTM syntax.

- This version of the package implements the following function:

```
rhythmtreetools.parse_rtm_syntax.parse_rtm_syntax()
```

- Example:

```
>>> from abjad.tools.rhythmtreetools import parse_rtm_syntax

>>> rtm = '(1 (1 (1 (1 1)) 1))'
>>> result = parse_rtm_syntax(rtm)
>>> result
FixedDurationTuplet(1/4, [c'8, c'16, c'16, c'8])
```

- Use the `rhythmtreetools` package to turn nested lists of numbers into Abjad tuplets.

Added new `timetokentools` package.

- This version of the package contains the following concrete classes:

```
timetokentools.NoteFilledTimeTokenMaker
timetokentools.OutputBurnishedSignalFilledTimeTokenMaker
timetokentools.OutputIncisedNoteFilledTimeTokenMaker
timetokentools.OutputIncisedRestFilledTimeTokenMaker
timetokentools.RestFilledTimeTokenMaker
timetokentools.SignalFilledTimeTokenMaker
timetokentools.TokenBurnishedSignalFilledTimeTokenMaker
timetokentools.TokenIncisedNoteFilledTimeTokenMaker
timetokentools.TokenIncisedRestFilledTimeTokenMaker
```

- The `timetokentools` package implements a family of related rhythm-making classes.
- Class usage follows a two-step initialize-then-call pattern.

Added new classes to `instrumenttools`.

- Added human voice classes:

```
instrumenttools.BaritoneVoice
instrumenttools.BassVoice
instrumenttools.ContraltoVoice
instrumenttools.MezzoSopranoVoice
instrumenttools.SopranoVoice
instrumenttools.TenorVoice
```

Added new time-interval tree functionality:

- Extended `TimeIntervalTree` with the following public methods:

```
scale_by_rational()
scale_to_rational()
shift_by_rational()
shift_to_rational()
split_at_rationals()
```

- These methods allow time-interval trees to behave more similar to time-intervals.

All score components are now public.

- The following classes are now publically available for the first time:

```
componenttools.Component
contexttools.Context
leaftools.Leaf
```

Further new functionality:

- Added the `marktools.BendAfter` class to model LilyPond's `\bendAfter` command:

```
>>> n = Note(0, 1)
>>> marktools.BendAfter(8)(n)
BendAfter(8.0)(c'1)
>>> f(n)
c'1 - \bendAfter #'8.0
```

- Added public pair property to `contexttools.TimeSignatureMark`:

```
>>> time_signature = contexttools.TimeSignatureMark((3, 16))
>>> time_signature.pair
(3, 16)
```

- Added `is_hairpin_token()` to `spannertools.HairpinSpanner` class.

Hairpin tokens are triples of the form `(x, y, z)` with dynamic tokens `x`, `y` and hairpin shape string `z`. For example `('p', '<', 'f')`.

- Added `resttools.replace_leaves_in_expr_with_rests()`.
- Added `leaftools.replace_leaves_in_expr_with_parallel_voices()`.
- Added `leaftools.replace_leaves_in_expr_with_named_parallel_voices()`.

Use the functions listed above to replace leaves in an expression with parallel voices containing copies of those leaves in both voices. This is useful for generating stemmed-glissandi structures.

- Added `contexttools.list_clef_names()`:

```
>>> contexttools.list_clef_names()
['alto', 'baritone', 'bass', 'mezzosoprano', 'percussion', 'soprano', 'treble']
```

- Added `find-slots-implementation-inconsistencies` development script.

Changes to end-user functionality:

- Changed `intervaltreertools` to `timeintervaltools`.
- Changed `contexttools.Context.context` to `contexttools.Context.context_name`.
- Calling `bool(Container())` on empty containers now returns `false`. The previous behavior of the system was to return `true`. The new behavior better conforms to the Python iterable interface.
- Moved `abjad/docs/scr/make-abjad-api` to `abjad/scr/make-abjad-api`.

1.3.4 Abjad 2.7

Released 2012-02-27. Built from r5100. Implements 221 public classes and 1029 functions totalling 168,000 lines of code.

- Added `lilypondparsertools.LilyPondParser` class, which parses a subset of LilyPond input syntax:

```
>>> from abjad.tools.lilypondparsertools import LilyPondParser
>>> parser = LilyPondParser()
>>> input = r"\new Staff { c'4 ( d'8 e' fs'2) \fermata }"
>>> result = parser(input)
>>> f(result)
\new Staff {
  c'4 (
    d'8
    e'8
```

```

    fs'2 -\fermata )
}

```

LilyPondParser defaults to English note names, but any of the other languages supported by LilyPond may be used:

```

>>> parser = LilyPondParser('nederlands')
>>> input = '{ c des e fis }'
>>> result = parser(input)
>>> f(result)
{
    c4
    df4
    e4
    fs4
}

```

Briefly, LilyPondParser understands theses aspects of LilyPond syntax:

- Notes, chords, rests, skips and multi-measure rests
- Durations, dots, and multipliers
- All pitchnames, and octave ticks
- Simple markup (i.e. `c'4 ^ "hello!"`)
- Most articulations
- Most spanners, including beams, slurs, phrasing slurs, ties, and glissandi
- Most context types via `\new` and `\context`, as well as context ids (i.e. `\new Staff = "foo" { }`)
- Variable assignment (i.e. `global = { \time 3/4 } \new Staff { \global }`)
- Many music functions: `- \acciaccatura - \appoggiatura - \bar - \breathe - \clef - \grace - \key - \transpose - \language - \makeClusters - \mark - \oneVoice - \relative - \skip - \slashedGrace - \time - \times - \transpose - \voiceOne, \voiceTwo, \voiceThree, \voiceFour`

LilyPondParser currently **DOES NOT** understand many other aspects of LilyPond syntax:

- `\markup`
 - `\book, \bookpart, \header, \layout, \midi and \paper`
 - `\repeat and \alternative`
 - Lyrics
 - `\chordmode, \drummode or \figuremode`
 - Property operations, such as `\override, \revert, \set, \unset, and \once`
 - Music functions which generate or extensively mutate musical structures
 - Embedded Scheme statements (anything beginning with `#`)
- Added `iotools.p()`, for conveniently parsing LilyPond syntax:

```

>>> result = p(r"\new Staff { c'4 d e f }")
>>> f(result)
\new Staff {
    c'4
}

```

```

    d4
    e4
    f4
}

```

- Added `schemetools.Scheme`, as a more robust replacement for nearly all other `schemetools` classes:

```

>>> from abjad.tools.schemetools import Scheme
>>> print Scheme(True).format
##t
>>> print Scheme('a', 'list', 'of', 'strings').format
#(a list of strings)
>>> print Scheme(('simulate', 'a', 'vector'), quoting="##").format
##(simulate a vector)
>>> print Scheme('a', ('nested', ('data', 'structure'))).format
#(a (nested (data structure))

```

- Removed deprecated `schemetools` classes:

- `SchemeBoolean`
- `SchemeFunction`
- `SchemeNumber`
- `SchemeString`
- `SchemeVariable`

In all cases, simply use `schemetools.Scheme` instead.

- Reimplemented `MarkupCommand`.

The new implementation is initialized from a command-name, and a variable-size list of arguments. Arguments which are lists or tuples will be enclosed in curly-braces:

```

>>> from abjad.tools.markuptools import MarkupCommand
>>> bold = MarkupCommand('bold', ['two', 'words'])
>>> rotate = MarkupCommand('rotate', 60, bold)
>>> triangle = MarkupCommand('triangle', False)
>>> concat = MarkupCommand('concat', ['one word', rotate, triangle])
>>> print concat.format
\concat { #"one word" \rotate #60 \bold { two words } \triangle ##f }

```

- Added `contexttools.TempoMarkInventory`, which models an ordered list of tempo marks:

```

>>> contexttools.TempoMarkInventory([('Andante', Duration(1, 8), 72), ('Allegro', Duration(1, 8), 120)])
TempoMarkInventory([TempoMark('Andante', Duration(1, 8), 72), TempoMark('Allegro', Duration(1, 8), 120)])

```

Inherits from `list`. Allows initialization, `append` and `extend` on tempo mark tokens.

- Added new `pitchtools.PitchRangeInventory` class.

The class acts as an ordered list of `PitchRange` objects.

The purpose of the class is to model something like palettes of different pitches available in all part of a score:

```

>>> pitchtools.PitchRangeInventory(['[C3, C6]', '[C4, C6]'])
PitchRangeInventory([PitchRange('[C3, C6]'), PitchRange('[C4, C6]')])

```

The class inherits from `list`.

- Added `sequencetools.all_are_pairs()` predicate:

```
>>> from abjad.tools.sequencetools import all_are_pairs
>>> all_are_pairs([(1, 2), (3, 4), (5, 6)])
True
```

- Added `sequencetools.all_are_pairs_of_types()` predicate:

```
>>> from abjad.tools.sequencetools import all_are_pairs_of_types
>>> all_are_pairs_of_types([(‘a’, 1.4), (‘b’, 2.3), (‘c’, 1.5)], str, float)
True
```

- Added `stringtools.is_underscore_delimited_lowercase_file_name_with_extension()` string predicate:

```
>>> stringtools.is_underscore_delimited_lowercase_file_name_with_extension(‘foo_bar.blah’)
True
```

- Added `iotools.is_underscore_delimited_file_name()` string predicate.

Returns true on any underscore-delimited lowercase string.

Also returns true on an underscore-delimited lowercase string terminated with an extension.

```
>>> stringtools.is_underscore_delimited_lowercase_file_name(‘foo_bar.py’)
True
```

```
>>> stringtools.is_underscore_delimited_lowercase_file_name(‘foo_bar’)
True
```

- Added `ImpreciseTempoError` to exceptions.
- Added `LilyPondParserError` to exceptions.
- Added `scr/devel/fix-test-cases`. The script is a two-line wrapper around the following other two scripts:

- `scr/devel/fix-test-case-names`
- `scr/devel/fix-test-case-numbers`

- Extended `Container` to use `LilyPondParser` to parse input strings.

- Extended `contexttools.InstrumentMark`, `scoretools.Performer` and `markuptools.Markup` with `__hash__` equality.

Now, if two instances compare equally (via `==`), their hashes also compare equally, allowing for more intuitive use of these classes as dictionary keys.

- Extended `contexttools.TempoMark` with textual indications and tempo ranges You may instantiate as normal, or in some new combinations:

```
>>> from abjad.tools.contexttools import TempoMark
>>> t = TempoMark(‘Langsam’, Duration(1, 4), (52, 57))
>>> t = TempoMark(‘Langsam’)
>>> t = TempoMark((1, 4), (52, 57))
```

In addition to its new read/write “textual_indication” attribute, `TempoMark` now also exposes a read-only “is_imprecise” property, which returns `True` if the mark cannot be expressed simply as `duration=units_per_minute`. Arithmetic operations on `TempoMarks` will now raise `ImpreciseTempoErrors` if any mark involved is imprecise.

- Extended tempo marks to be able to initialize from ‘tempo mark tokens’. A tempo mark token is a length-2 or length-3 tuple of tempo mark arguments.

- Extended tempo mark with `is_tempo_mark_token()` method:

```
>>> tempo_mark = contexttools.TempoMark(Duration(1, 4), 72)
>>> tempo_mark.is_tempo_mark_token((Duration(1, 4), 84))
True
```

- Extended case-testing `iotools` string predicates to allow digits.

Functions changed:

- `stringtools.is_space_delimited_lowercase_string`
- `stringtools.is_underscore_delimited_lowercase_file_name`
- `stringtools.is_lowercamelcase_string`
- `stringtools.is_uppercamelcase_string`
- `stringtools.is_underscore_delimited_lowercase_string`
- `stringtools.is_underscore_delimited_lowercase_file_name_with_extension`

- Extended `lilypondfiletools.NonattributedBlock` with `is_formatted_when_empty` read-write property. `lilypondfiletools.ScoreBlock` no longer formats when empty, by default.
- Extended `marktools.BarLine` with `format_slot` keyword.
- Extended `pitchtools.PitchRange` class with read-only `pitch_range_name` and `pitch_range_name_markup` attributes.
- Extended `scoretools.InstrumentationSpecifier` with read-only `performer_name_string` attribute.
- Extended all `beamtools.Beam`-, `Slur`- and `Hairpin`-related spanner classes, as well as `tietools.TieSpanner` with an optional `direction` keyword:

```
>>> c = Container("c'4 d'4 e'4 f'4")
>>> spanner = spannertools.SlurSpanner(c[:], 'up')
>>> f(c)
{
    c'4 ^ (
    d'4
    e'4
    f'4 )
}
```

The direction options are exactly the same as for `Articulation` and `Markup`: `'up'`, `'^'`, `'down'`, `'_'`, `'neutral'`, `'-'` and `None`.

- Extended `tonalitytools.Scale` with `create_named_chromatic_pitch_set_in_pitch_range()` method.
- Changed `tuplettools.FixedDurationTuplet.multiplier` to return fraction instead of duration.
- Renamed attributes, methods and functions throughout `intervalreetools`:
 - `centroid` => `center` (except where a weighted mean is actually used)
 - `high` => `stop`
 - `high_min` => `earliest_stop`
 - `high_max` => `latest_stop`
 - `low` => `start`
 - `low_min` => `earliest_start`

- low_max => latest_start
- magnitude => duration

This both clarifies the API, and prevents shadowing of Python's builtin `min()` and `max()`.

- Renamed `marktools.Articulation.direction_string` => `marktools.Articulation.direction`.
- Renamed `markuptools.Markup.direction_string` => `'markuptools.Markup.direction`.
- Renamed `tuplettools.Tuplet.ratio` to `tuplettools.Tuplet.ratio_string`.
- Renamed `scr/devel/find-nonalphabetized-method-names` to `scr/devel/find-nonalphabetized-class-attributes`.
- Improved `scr/devel/find-nonalphabetized-methods`.
- Updated literature examples to match API changes.
- Removed ancient `stafftools.make_invisible_staff()`.
- Added `text_editor` key to user config dictionary (in `~/abjad/config.py`).
- Improved `__repr__` strings of `tonalitytools.Mode` and `tonalitytools.Scale`.
- `contexttools.TempoMark.__repr__` now shows `__repr__` version of duration instead of string version of duration.
- `scr/devel/abj-grp` no longer excludes lines of code that include the string `'svn'`.

1.3.5 Abjad 2.6

Released 2012-01-29. Built from r4979. Implements 197 public classes and 941 public functions totalling 153,000 lines of code.

- Added top-level `decorators` directory with `requires` decorator. The `requires` decorator renders the following two function definitions equivalent:

```
from abjad.tools.decoratortools import requires

@requires(int)
def foo(x):
    return x ** 2

def foo(x):
    assert isinstance(x, int)
    return x ** 2
```

- Added new classes to `scoretools`:

```
scoretools.InstrumentationSpecifier
scoretools.Performer
```

- Added `scoretools.list_performer_names()`:

```
>>> for name in scoretools.list_performer_names()[:10]:
...     name
...
'accordionist'
'bassist'
'bassoonist'
'cellist'
'clarinetist'
```

```
'flutist'
'guitarist'
'harpist'
'harpsichordist'
'hornist'
```

- Added `scoretools.list_primary_performer_names()`.
- Added `measuretools.measure_to_one_line_input_string()`:

```
>>> measure = Measure((3, 4), "c4 d4 e4")

>>> measure
Measure(3/4, [c4, d4, e4])

>>> measuretools.measure_to_one_line_input_string(measure)
"Measure((3, 4), 'c4 d4 e4')"
```

- Added new classes to `instrumenttools`:

```
SopraninoSaxophone
SopranoSaxophone
AltoSaxophone
TenorSaxophone
BaritoneSaxophone
BassSaxophone
ContrabassSaxophone

ClarinetInA

AltoTrombone
BassTrombone

Harpsichord
```

- Added known untuned percussion:

```
>>> for name in instrumenttools.UntunedPercussion.known_untuned_percussion[:10]:
...     print name
...
agogô
anvil
bass drum
bongo drums
cabasa
cajón
castanets
caxixi
claves
conga drums
```

- Added `_Instrument.get_default_performer_name()`:

```
>>> bassoon = instrumenttools.Bassoon()

>>> bassoon.get_default_performer_name()
'bassoonist'
```

- Added `_Instrument.get_performer_names()`:

```
>>> bassoon.get_performer_names()
['instrumentalist', 'reed player', 'double reed player', 'bassoonist']
```

- Added read / write `_Instrument.pitch_range`:

```
>>> marimba.pitch_range = (-24, 36)
>>> marimba.pitch_range
PitchRange(' [C2, C7]')
```

- Added read-only `_Instrument.traditional_pitch_range`:

```
>>> marimba = instrumenttools.Marimba()
>>> marimba.traditional_pitch_range
PitchRange(' [F2, C7]')
```

- Added `instrumenttools.list_instruments()`:

```
>>> for instrument_name in instrumenttools.list_instrument_names()[:10]:
...     instrument_name
...
'accordion'
'alto flute'
'alto saxophone'
'alto trombone'
'clarinet in B-flat'
'baritone saxophone'
'bass clarinet'
'bass flute'
'bass saxophone'
'bass trombone'
```

- Added other functions to `instrumenttools`:

```
instrumenttools.list_primary_instrument_names()
instrumenttools.list_secondary_instrument_names()
```

- Added new class to `lilypondfiletools`:

```
ContextBlock
```

- Added `pitchtools.is_symbolic_pitch_range_string()`:

```
>>> pitchtools.is_symbolic_pitch_range_string(' [A0, C8]')
True
```

- Added `pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name()`:

```
>>> pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name('A#4')
'as' "
```

- Added `pitchtools.symbolic_accidental_string_to_alphabetic_accidental_string_abbreviation`:

```
>>> pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string('tqs')
'#+' "
```

- Added other new functions to `pitchtools`:

```
pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string()
pitchtools.is_symbolic_accidental_string()
pitchtools.is_pitch_class_octave_number_string()
```

- Added `stringtools.string_to_strict_directory_name()`:

```
>>> stringtools.string_to_strict_directory_name('Déja vu')
'deja_vu'
```

- Added `stringtools.strip_diacritics_from_binary_string()`:

```
>>> binary_string = 'Dvořák'
>>> stringtools.strip_diacritics_from_binary_string(binary_string)
'Dvorak'
```

- Added other new functions to `iotools`:

```
stringtools.capitalize_string_start()
iotools.is_space_delimited_lowercamelcase_string()
iotools.is_underscore_delimited_lowercamelcase_package_name()
iotools.is_underscore_delimited_lowercamelcase_string()
stringtools.is_lowercamelcase_string()
stringtools.is_uppercamelcase_string()
stringtools.space_delimited_lowercase_to_uppercamelcase()
stringtools.uppercamelcase_to_space_delimited_lowercase()
stringtools.uppercamelcase_to_underscore_delimited_lowercase()
```

- Added new functions to `mathtools`:

```
mathtools.is_positive_integer_power_of_two()
mathtools.is_integer_equivalent_expr()
```

- Added sequence type-checking predicates:

```
chordtools.all_are_chords()
containertools.all_are_containers()
durationtools.all_are_duration_tokens()
durationtools.all_are_durations()
gracetools.all_are_grace_containers()
leaftools.all_are_leaves()
markuptools.all_are_markup()
measuretools.all_are_measures()
notetools.all_are_notes()
pitcharraytools.all_are_pitch_arrays()
pitchtools.all_are_named_chromatic_pitch_tokens()
resttools.all_are_rests()
scoretools.all_are_scores()
sievetoools.all_are_residue_class_expressions()
skiptools.all_are_skips()
spannertools.all_are_spanners()
stafftools.all_are_staves()
tupletoools.all_are_tuplets()
```

- Extended `NamedChromaticPitch` to allow initialization from pitch-class / octave number strings:

```
>>> pitchtools.NamedChromaticPitch('C#2')
NamedChromaticPitch('cs,')
```

- Extended `PitchRange` to allow initialization from symbolic pitch range strings:

```
>>> pitchtools.PitchRange(' [A0, C8]')
PitchRange(' [A0, C8]')
```

- Extended `PitchRange` to allow initialization from pitch-class / octave number strings:

```
>>> pitchtools.PitchRange('A0', 'C8')
PitchRange('[A0, C8]')
```

- Extended `leaftools.is_bar_line_crossing_leaf()` to work when no explicit time signature mark is found.
- Extended Markup to be able to function as a top-level `LilyPondFile` element.
- Extended instruments with `is_primary` and `is_secondary` attributes.
- Extended instruments with `instrument_name` and `instrument_name_markup` attributes.
- Extended instruments with `short_instrument_name` and `short_instrument_name_markup` attributes.
- Extended `iotools.write_expr_to_ly()` and `iotools.write_expr_to_pdf()` with `'tagline'` keyword.
- Extended `replace-in-files` script to skip `.text`, `.ly` and `.txt` files.
- Renamed `Accidental.symbolic_string` to `Accidental.symbolic_accidental_string`.
- Renamed `Accidental.alphabetic_string` to `Accidental.alphabetic_accidental_abbreviation`.
- Fixed bug in `iotools.play()`.
- Fixed bug in `quantizationtools` regarding quantizing a stream of `QEvents` directly.

1.3.6 Abjad 2.5

Released 2011-09-22. Built from r4803.

- Added `get_leaf_in_expr_with_minimum_prolated_duration()` function to `leaftools`.
- Added `get_leaf_in_expr_with_maximum_prolated_duration()` function to `leaftools`.
- Added `are_relatively_prime()` function to `mathtools`.
- Added `CyclicTree` class to `sequencetools`.
- Added `get_next_n_nodes_at_level(n, level)` method to `sequencetools.Tree`.
- Extended `spanners` to sort by repr.
- Renamed `lilyfiletools` to `lilypondfiletools`.
- Renamed `lilyfiletools.LilyFile` to `lilypondfiletools.LilyPondFile`.
- Renamed `lilyfiletools.make_basic_lily_file()` to `lilypondfiletools.make_basic_lilypond_file`.

Note that the three renames change user syntax. Composers working with the `lilypondfiletools` module should update their score code.

1.3.7 Abjad 2.4

Released 2011-09-12. Built from r4769.

- Added Mozart *Musikalisches Würfelspiel*.

Ein Musikalisches Wuerfelspiel



- Added new Tree class to sequencetools to work with sequences whose elements have been grouped into arbitrarily many levels of containment.
- Added new BarLine class to marktools package.
- Added new HorizontalBracketSpanner to spannertools package.
- Improved schemetools.SchemePair handling.
- Extended LilyPondFile blocks with double underscore-delimited attributes.

1.3.8 Abjad 2.3

Released 2011-09-04. Built from r4747.

Filled out the API for working with marks:

```
marktools.attach_articulations_to_components_in_expr()
marktools.detach_articulations_attached_to_component()
marktools.get_articulations_attached_to_component()
marktools.get_articulation_attached_to_component()
marktools.is_component_with_articulation_attached()
```

These five type of functions are now implemented for the following marks:

```
marktools.Annotation
marktools.Articulation
marktools.LilyPondCommandMark
marktools.LilyPondComment
marktools.StemTremolo
```

The same type of functions are likewise implemented for the following context marks:

```
contexttools.ClefMark
contexttools.DynamicMark
contexttools.InstrumentMark
contexttools.KeySignatureMark
contexttools.StaffChangeMark
contexttools.TempoMark
contexttools.TimeSignatureMark
```

- Extended Container.extend() to allow for LilyPond input strings. You can now say container.extend("c'4 d'4 e'4 f'4").
- Added public parent attribute to all components. You can now say note.parent. The attribute is read-only.

- Added `cfgtools.list_package_dependency_version()`.
- Added `py.test` and Sphinx dependencies to the Abjad package.
- Added LilyPond command mark chapter to reference manual
- Renamed `cfgtools` to `configurationtools`.
- Renamed `durtools` to `durationtools`.
- Renamed `metertools` to `timesignaturetools`.
- Renamed `seqtools` to `sequencetools`.
- Renamed `Mark.attach_mark()` to `Mark.attach()`.
- Renamed `Mark.detach_mark()` to `Mark.detach()`.
- Renamed `marktools.Comment` to `marktools.LilyPondComment`. This matches `marktools.LilyPondCommandMark`.
- Removed `contexttools.TimeSignatureMark(3, 8)` initialization. You must now say `contexttools.TimeSignatureMark((3, 8))` instead. This parallels the initialization syntax for rests, skips and measures.

1.3.9 Abjad 2.2

Released 2011-08-30. Built from r4677.

- Added articulations chapter to reference manual.
- Reordered the way in which Abjad determines the value of the `HOME` environment variable.
- Updated `scr/devel/replace-in-files` to avoid image files.
- Updated `iotools.log()` to call operating-specific text editor.

1.3.10 Abjad 2.1

Released 2011-08-21. Built from r4655.

- Updated instrument mark `repr` to display target context when instrument mark is attached.
- Extended `scr/abj` and `scr/abjad` to display Abjad version and revision numbers on startup.

1.3.11 Abjad 2.0

Released 2011-08-17. Built from r4638.

Abjad 2.0 is the first public release of Abjad in more than two years. The new release of the system more than doubles the number of classes, functions and packages available in Abjad.

- The API has been cleaned up and completely reorganized. Features have been organized into a collection of 39 different libraries:

<code>cfgtools/</code>	<code>instrumenttools/</code>	<code>mathtools/</code>	<code>resttools/</code>	<code>tempotools/</code>
<code>chordtools/</code>	<code>intervaltreetools/</code>	<code>measuretools/</code>	<code>schemetools/</code>	<code>threadtools/</code>
<code>componenttools/</code>	<code>iotools/</code>	<code>metertools/</code>	<code>scoretools/</code>	<code>tietools/</code>
<code>containertools/</code>	<code>layouttools/</code>	<code>musicxmltools/</code>	<code>seqtools/</code>	<code>tonalitytools/</code>
<code>contexttools/</code>	<code>leaftools/</code>	<code>notetools/</code>	<code>sievetools/</code>	<code>tuplettools/</code>
<code>durtools/</code>	<code>lilyfiletools/</code>	<code>pitcharraytools/</code>	<code>skiptools/</code>	<code>verticalitytools/</code>

<code>gracetools/</code>	<code>marktools/</code>	<code>pitchtools/</code>	<code>spannertools/</code>	<code>voicetools/</code>
<code>importtools/</code>	<code>markuptools/</code>	<code>quantizationtools/</code>	<code>stafftools/</code>	

- The name of almost every function in the public API has been changed to better indication what the function does. While this has the effect of making Abjad 2.0 largely non-backwards compatible with code written in Abjad 1.x, the longer and much more explicit function names in Abjad 2.0 make code used to structure complex scores dramatically easier to maintain and understand.
- The `contexttools`, `instrumenttools`, `intervaltreertools`, `lilyfiletools`, `marktools`, `pitcharraytools`, `quantizationtools`, `sievetools`, `tonalitytools` and `verticalitytools` packages are completely new.
- The classes implemented in the `contexttools` and `marktools` packages provide an object-oriented interfaces to clefs, time signatures, key signatures, articulations, tempo marks and other symbols stuck to the outside of the hierarchical score tree. The classes implemented in `contexttools` and `marktools` model information outside the score tree much the way that the classes implemented in `spannertools` implement object-oriented interfaces to beams, brackets, hairpins, glissandi and other line-like symbols.
- The `instrumenttools` package provides an object-oriented model of most of the conventional instruments of the orchestra.
- The `intervaltreertools` package implements a custom way of working with chunks of score during composition.
- The `lilyfiletools` package implements an object-oriented interface to arbitrarily structured LilyPond input files.
- The `pitcharraytools` package implements an object-oriented way of composing with pitches, pitch-classes and other pitch-related objects independent of rhythmic context.
- The experimental `quantizationtools` package implements classes and functions for quantizing rhythmic events.
- The `sievetools` package implements an object-oriented interface to the basics of Xenakis's system of sieves.
- The `tonalitytools` package implements classes and methods to model the basics of functional harmonic analysis.
- The `verticalitytools` package provides vertical-moment-based iteration and analysis of any score.
- The `pitchtools` package has grown considerably in size and functionality. Classes now exist to model named and numbered chromatic pitches (and pitch-classes), named and numbered diatonic pitches (and pitch-classes), melodic and harmonic diatonic intervals (and interval-classes), melodic and harmonic chromatic intervals (and interval-classes), as well as ordered segments and unordered sets of these and related objects. The package contains dozens of functions to create, inspect, iterate, analyze and transpose these classes and their collections.
- The old `listtools` package has been renamed `seqtools`.
- Dozens of new functions for cutting, pasting, partitioning, breaking, arranging and reordering score components have been added to the system. See the new functions in `componenttools`, `containertools`, `leaftools`, `measuretools` and `scoretools` for details.
- The core component classes modeling notes, rests, chords, tuplets, measures, voices, staves and scores have been reimplemented to consume dramatically less memory, making it much easier to work with arrays of hundreds and thousands of components.
- Abjad core formatting logic has been optimized to make the formatting of scores with hundreds or thousands of events take much less time than before.
- The component duration interfaces have been replaced by more straightforward read-only component attributes.
- Added Ferneyhough Unsichbare Farben example.

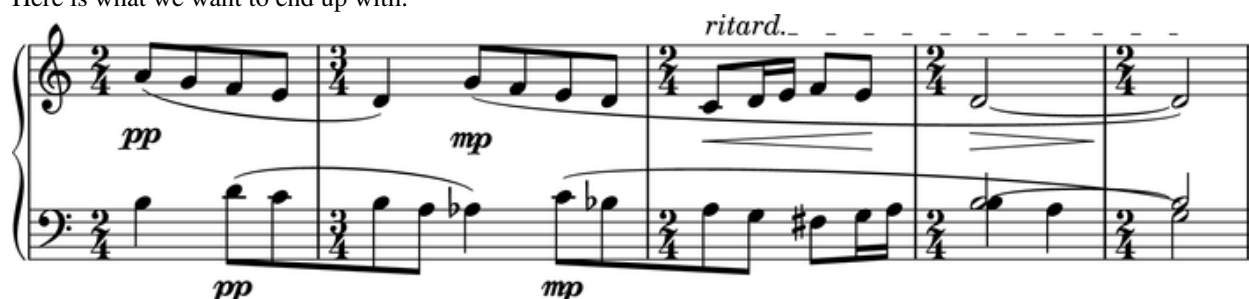
The musical score consists of 10 staves, each containing a sequence of notes with various rhythmic groupings indicated by brackets and ratios. The ratios include 3:2, 5:4, 7:4, 9:8, 11:8, 12:11, 18:11, and 15:11. The notes are primarily eighth and sixteenth notes, often beamed together in groups.

EXAMPLES

2.1 Bartók: *Mikrokosmos*

This example reconstructs the last five measures of Bartók's "Wandering" from *Mikrokosmos*, volume III. The end result is just a few measures long but covers the basic features you'll use most often in Abjad.

Here is what we want to end up with:



2.1.1 The score

We'll construct the fragment top-down from containers to notes. We could have done it the other way around but it will be easier to keep the big picture in mind this way. Later, you can rebuild the example bottom-up as an exercise.

First let's create an empty score with a pair of staves connected by a brace:

```
>>> score = Score([])
>>> piano_staff = scoretools.PianoStaff([])
>>> upper_staff = Staff([])
>>> lower_staff = Staff([])

>>> piano_staff.append(upper_staff)
>>> piano_staff.append(lower_staff)
>>> score.append(piano_staff)
```

Here we create an empty score and assign it to the `score` variable. Then we create an empty piano staff assigned to the `piano_staff` variable and two empty staves assigned to the `upper_staff` and `lower_staff` variables. Finally, we append the two staves to the piano staff and the piano staff to the score.

2.1.2 The measures

Now let's add some measures to our score:

```
>>> m1 = Measure((2, 4), [])
>>> m2 = Measure((3, 4), [])
>>> m3 = Measure((2, 4), [])
>>> m4 = Measure((2, 4), [])
>>> m5 = Measure((2, 4), [])

>>> upper_measures = [m1, m2, m3, m4, m5]
>>> lower_measures = componenttools.copy_components_and_covered_spanners(upper_measures)

>>> upper_staff.extend(upper_measures)
>>> lower_staff.extend(lower_measures)
```

The lower measures are copies of the upper measures.

Note that we add lists of measures to staves with `extend()`. This is because `extend()` is used for adding many objects to an iterable at once while `append()` is used to add only one object at a time.

2.1.3 The notes

Now let's add some notes. We begin with the upper staff:

```
>>> upper_measures[0].extend([Note(i, (1, 8)) for i in [9, 7, 5, 4]])

>>> upper_measures[1].extend(notetools.make_notes([2, 7, 5, 4, 2], [(1, 4)] + [(1, 8)] * 4))

>>> notes = notetools.make_notes([0, 2, 4, 5, 4], [(1, 8), (1, 16), (1, 16), (1, 8), (1, 8)])
>>> upper_measures[2].extend(notes)

>>> upper_measures[3].append(Note("d'2"))

>>> upper_measures[4].append(Note("d'2"))
```

Now let's add notes to the lower staff. This will be a more intricate process than that needed for the upper staff. We added notes directly to the measures of the upper staff. But this will not be possible for the lower staff because of the simultaneous voices the lower staff contains.

We add notes to the lower staff measure by measure:

```
>>> main_voice_m1 = Voice("b4 d'8 c'8")
>>> main_voice_m1.name = 'main_voice'
>>> lower_measures[0].append(main_voice_m1)

>>> main_voice_m2 = Voice("b8 a8 af4 c'8 bf8")
>>> main_voice_m2.name = 'main_voice'
>>> lower_measures[1].append(main_voice_m2)

>>> main_voice_m3 = Voice("a8 g8 fs8 g16 a16")
>>> main_voice_m3.name = 'main_voice'
>>> lower_measures[2].append(main_voice_m3)
```

Notice that we give the same name to the three voices contained in the first three measures of the lower staff.

It is in the last two measures of the lower staff where Bartók writes two voices at once. We'll name the second of these two voices the *appendix_voice*:

```
>>> appendix_voice_m4 = Voice([Note("b2")])
>>> appendix_voice_m4.name = 'appendix_voice'
>>> lilypond_command_mark = marktools.LilyPondCommandMark('voiceOne')
```

```
>>> lilypond_command_mark.attach(appendix_voice_m4)
LilyPondCommandMark('voiceOne') (Voice-"appendix_voice"{1})

>>> main_voice_m4 = Voice("b4 a4")
>>> main_voice_m4.name = 'main_voice'
>>> lilypond_command_mark = marktools.LilyPondCommandMark('voiceTwo')
>>> lilypond_command_mark.attach(main_voice_m4)
LilyPondCommandMark('voiceTwo') (Voice-"main_voice"{2})

>>> container = Container([appendix_voice_m4, main_voice_m4])
>>> container.is_parallel = True
>>> lower_measures[3].append(container)
```

The LilyPond `\voiceOne` and `\voiceTwo` commands determine the direction of the stems in different voices.

Note that we must put both voices in a parallel container because they occur at the same time in the score. We do this by creating an Abjad container and then setting the `is_parallel` attribute of the container to true.

We now do a similar thing for the last measure:

```
>>> appendix_voice_m5 = Voice("b2")
>>> appendix_voice_m5.name = 'appendix_voice'
>>> lilypond_command_mark = marktools.LilyPondCommandMark('voiceOne')
>>> lilypond_command_mark.attach(appendix_voice_m5)
LilyPondCommandMark('voiceOne') (Voice-"appendix_voice"{1})

>>> main_voice_m5 = Voice("g2")
>>> main_voice_m5.name = 'main_voice'
>>> lilypond_command_mark = marktools.LilyPondCommandMark('voiceTwo')
>>> lilypond_command_mark.attach(main_voice_m5)
LilyPondCommandMark('voiceTwo') (Voice-"main_voice"{1})

>>> container = Container([appendix_voice_m5, main_voice_m5])
>>> container.is_parallel = True
>>> lower_measures[4].append(container)
```

Here's our work so far:

```
>>> show(score)
```



2.1.4 The details

Ok, let's add the details. First, notice that the bottom staff has a treble clef just like the top staff. Let's change that:

```
>>> contexttools.ClefMark('bass') (lower_staff)
ClefMark('bass') (Staff{5})
```

Now let's add dynamic marks. For the top staff, we'll add them to the first note of the first measure and the second note of the second measure. For the bottom staff, we'll add dynamic markings to the second note of the first measure and the fourth note of the second measure.

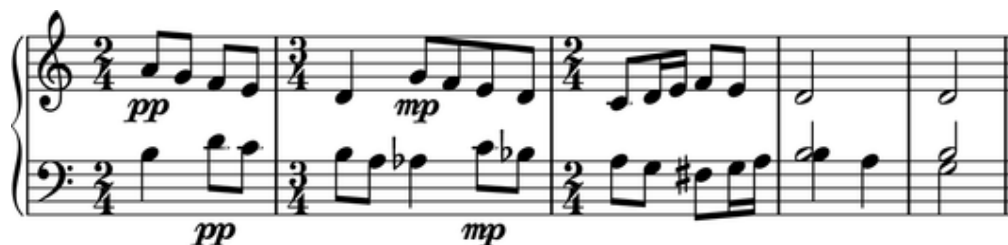
```
>>> contexttools.DynamicMark('pp')(upper_measures[0][0])
DynamicMark('pp')(a'8)
>>> contexttools.DynamicMark('mp')(upper_measures[1][1])
DynamicMark('mp')(g'8)
>>> contexttools.DynamicMark('pp')(lower_measures[0][0][1])
DynamicMark('pp')(d'8)
>>> contexttools.DynamicMark('mp')(lower_measures[1][0][3])
DynamicMark('mp')(c'8)
```

Let's add a double bar to the end of the piece:

```
>>> bar_line = marktools.BarLine('|.')
>>> bar_line.attach(lower_staff.leaves[-1])
BarLine('|.')(g2)
```

And see how things are coming out:

```
>>> show(score)
```



Notice that the beams of the eighth and sixteenth notes appear as you would usually expect: grouped by beat. We get this for free thanks to LilyPond's default beaming algorithm. But this is not the way Bartók notated the beams. Let's set the beams as Bartók did with some crossing the bar lines:

```
>>> beamtools.BeamSpanner(upper_measures[0])
BeamSpanner(|2/4(4)|)
>>> beamtools.BeamSpanner(lower_staff.leaves[1:5])
BeamSpanner(d'8, c'8, b8, a8)
>>> beamtools.BeamSpanner(lower_staff.leaves[6:10])
BeamSpanner(c'8, bf8, a8, g8)
```

```
>>> show(score)
```



Now some slurs:

```
>>> spannertools.SlurSpanner(upper_staff.leaves[0:5])
SlurSpanner(a'8, g'8, f'8, e'8, d'4)
>>> spannertools.SlurSpanner(upper_staff.leaves[5:])
SlurSpanner(g'8, f'8, ... [7] ..., d'2, d'2)
```

```
>>> spannertools.SlurSpanner(lower_staff.leaves[1:6])
SlurSpanner(d'8, c'8, b8, a8, af4)
>>> spannertools.SlurSpanner(lower_staff.leaves[6:13] + (main_voice_m4, main_voice_m5))
SlurSpanner(c'8, bf8, ... [5] ..., {b4, a4}, {g2})
```

Hairpins:

```
>>> spannertools.CrescendoSpanner(upper_staff.leaves[-7:-2])
CrescendoSpanner(c'8, d'16, e'16, f'8, e'8)
>>> spannertools.DecrescendoSpanner(upper_staff.leaves[-2:])
DecrescendoSpanner(d'2, d'2)
```

A ritardando marking above the last seven notes of the upper staff:

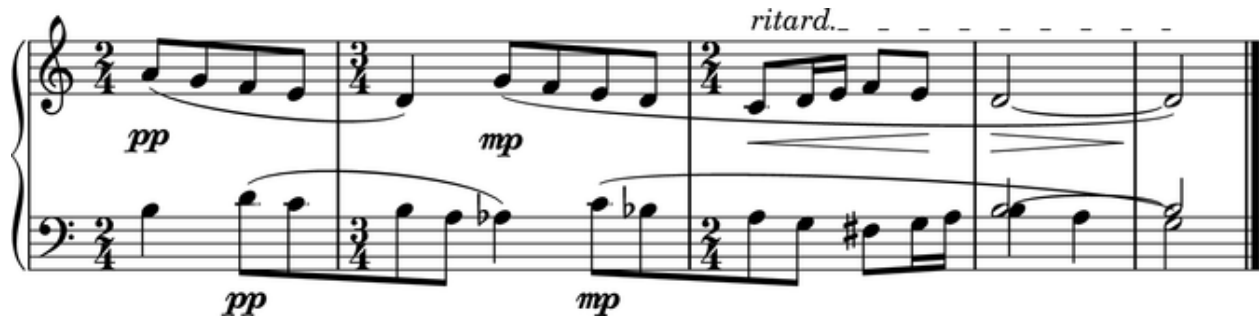
```
>>> text_spanner = spannertools.TextSpanner(upper_staff.leaves[-7:])
>>> text_spanner.override.text_spanner.bound_details__left__text = markuptools.Markup('ritard..')
```

And ties connecting the last two notes in each staff:

```
>>> tietools.TieSpanner(upper_staff[-2:])
TieSpanner(|2/4(1)|, |2/4(1)|)
>>> tietools.TieSpanner([appendix_voice_m4[0], appendix_voice_m5[0]])
TieSpanner(b2, b2)
```

The final result:

```
>>> show(score)
```



2.2 Ferneyhough: *Unsichtbare Farben*

Mikhail Malt analyzes the rhythmic materials of Ferneyhough’s *Unsichtbare Farben* in *The OM Composer’s Book 2*.

Malt explains that Ferneyhough used OpenMusic to create an “exhaustive catalogue of rhythmic cells” such that:

1. They are subdivided into two pulses, with proportions from 1/1 to 1/11.
2. The second pulse is subdivided successively by 1, 2, 3, 4, 5 and 6.

Let’s recreate Malt’s results in Abjad.

2.2.1 The proportions

First we define proportions:

```
>>> proportions = [(1, n) for n in range(1, 11 + 1)]
```

```
>>> proportions
[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (1, 11)]
```

2.2.2 The transforms

Then we make aliases to give shorter names to two functions with long names:

```
>>> make_tuplet = tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_encourage_dots
>>> tie_chain_to_tuplet = tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots
```

And then define a helper function:

```
def divide_tuplet(tuplet, n):
    last_tie_chain = tietools.get_tie_chain(tuplet[-1])
    proportions = n * [1]
    tie_chain_to_tuplet(last_tie_chain, proportions)
```

2.2.3 The rhythms

We set the duration of each tuplet equal to a quarter note:

```
>>> duration = Fraction(1, 4)
```

And then we make the rhythms:

```
>>> music = []
>>> for proportion in proportions:
...     tuplets = []
...     for n in range(1, 6 + 1):
...         tuplet = make_tuplet(duration, proportion)
...         divide_tuplet(tuplet, n)
...         tuplets.append(tuplet)
...     music.extend(tuplets)
...
...
```

2.2.4 The score

Finally we make the score:

```
>>> staff = stafftools.RhythmicStaff(music)
>>> score = Score([staff])
>>> lilypond_file = lilypondfiletools.make_basic_lilypond_file(score)
```

Configure containers:

```
>>> contexttools.TimeSignatureMark((1, 4))(staff)
TimeSignatureMark((1, 4))(RhythmicStaff{66})
>>> score.override.bar_number.transparent = True
>>> score.set.proportional_notation_duration = schemetools.SchemeMoment(1, 56)
>>> score.set.tuplet_full_length = True
>>> score.override.spacing_spanner.uniform_stretching = True
>>> score.override.spacing_spanner.strict_note_spacing = True
>>> score.override.tuplet_bracket.padding = 2
>>> score.override.tuplet_bracket.staff_padding = 4
>>> score.override.tuplet_number.text = schemetools.SchemeFunction('tuplet-number::calc-fraction-text')
```



```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
AttributeError: 'module' object has no attribute 'SchemeFunction'
>>> score.override.time_signature.stencil = False
>>> score.override.bar_line.stencil = False
```

Import layout tools:

```
>>> from abjad.tools import layouttools
```

Configure the LilyPond file:

```
>>> lilypond_file.default_paper_size = '11x17', 'portrait'
>>> lilypond_file.global_staff_size = 12
>>> lilypond_file.layout_block.indent = 0
>>> lilypond_file.layout_block.ragged_right = True
>>> lilypond_file.paper_block.ragged_bottom = True
>>> lilypond_file.paper_block.system_system_spacing = layouttools.make_spacing_vector(0, 0, 8, 0)
```

And show the result:

```
>>> show(lilypond_file)
```



2.3 Ligeti: *Désordre*

This example demonstrates the power of exploiting redundancy to model musical structure. The piece that concerns us here is Ligeti’s *Désordre*: the first piano study from Book I. Specifically, we will focus on modeling the first section of the piece:

ÉTUDE 1 : "DÉSORDRE" Dédicée à Pierre Boulez

György Ligeti
1985

Molto vivace, Vigoroso, molto ritmico ♩ = 76

Piano

f p pp fp fP sfz sfz sfz sfz sfz sfz sempre simile

sempre legatissimo possibile

f p pp fp fP sfz sfz sfz sfz sfz sfz sempre simile

f p pp fp fP sfz sfz sfz sfz sfz sfz sempre simile

Stets sehr sparsamer Gebrauch des Pedals / Utiliser la pédale très discrètement (pendant toute la pièce)

The redundancy is immediately evident in the repeating pattern found in both staves. The pattern is hierarchical. At the smallest level we have what we will here call a *cell*:



There are two of these cells per measure. Notice that the cells are strictly contained within the measure (i.e., there are no cells crossing a bar line). So, the next level in the hierarchy is the measure. Notice that the measure sizes (the meters) change and that these changes occur independently for each staff, so that each staff carries it's own sequence of measures. Thus, the staff is the next level in the hierarchy. Finally there's the piano staff, which is composed of the right hand and left hand staves.

In what follows we will model this structure in this order (*cell*, *measure*, *staff*, *piano staff*), from bottom to top.

2.3.1 The cell

Before plunging into the code, observe the following characteristic of the *cell*:

1. It is composed of two layers: the top one which is an octave “chord” and the bottom one which is a straight eighth note run.
2. The total duration of the *cell* can vary, and is always the sum of the eight note runs.
3. The eight note runs are always stem down while the octave “chord” is always stem up.
4. The eight note runs are always beamed together and slurred, and the first two notes always have the dynamic markings ‘f’ ‘p’.

The two “layers” of the *cell* we will model with two Voices inside a parallel Container. The top Voice will hold the octave “chord” while the lower Voice will hold the eighth note run. First the eighth notes:

```
>>> pitches = [1, 2, 3]
>>> notes = notetools.make_notes(pitches, [(1, 8)])
>>> beamtools.BeamSpanner(notes)
BeamSpanner(cs'8, d'8, ef'8)
>>> spannertools.SlurSpanner(notes)
SlurSpanner(cs'8, d'8, ef'8)
>>> contextttools.DynamicMark('f')(notes[0])
DynamicMark('f')(cs'8)
>>> contextttools.DynamicMark('p')(notes[1])
DynamicMark('p')(d'8)

>>> voice_lower = Voice(notes)
>>> voice_lower.name = 'rh_lower'
>>> marktools.LilyPondCommandMark('voiceTwo')(voice_lower)
LilyPondCommandMark('voiceTwo')(Voice-"rh_lower"{3})
```

The notes belonging to the eighth note run are first beamed and slurred. Then we add the dynamic marks to the first two notes, and finally we put them inside a Voice. After naming the voice we number it 2 so that the stems of the notes point down.

Now we construct the octave:

```
>>> import math
>>> n = int(math.ceil(len(pitches) / 2.))
>>> chord = Chord([pitches[0], pitches[0] + 12], (n, 8))
>>> marktools.Articulation('>')(chord)
Articulation('>')(<cs' cs''>4)

>>> voice_higher = Voice([chord])
>>> voice_higher.name = 'rh_higher'
>>> marktools.LilyPondCommandMark('voiceOne')(voice_higher)
LilyPondCommandMark('voiceOne')(Voice-"rh_higher"{1})
```

The duration of the chord is half the duration of the running eighth notes if the duration of the running notes is divisible by two. Otherwise the duration of the chord is the next integer greater than this half. We add the articulation marking and finally add the Chord to a Voice, to which we set the number to 1, forcing the stem to always point up.

Finally we combine the two voices in a parallel Container:

```
>>> p = Container([voice_lower, voice_higher])
>>> p.is_parallel = True
```

This results in the complete *Désordre cell*:

```
>>> cell = Staff([p])
```

Because this *cell* appears over and over again, we want to reuse this code to generate any number of these *cells*. We here encapsulate it in a function that will take only a list of pitches:

```
def desordre_cell(pitches):
    """The function constructs and returns a *Désordre cell*.
    'pitches' is a list of numbers or, more generally, pitch tokens.
    """
    notes = [Note(p, (1, 8)) for p in pitches]
    beamtools.BeamSpanner(notes)
    spannertools.SlurSpanner(notes)
    contextttools.DynamicMark('f')(notes[0])
```

```

contexttools.DynamicMark('p')(notes[1])
# make the lower voice
v_lower = Voice(notes)
v_lower.name = 'rh_lower'
marktools.LilyPondCommandMark('voiceTwo')(v_lower)
n = int(math.ceil(len(pitches) / 2.))
chord = Chord([pitches[0], pitches[0] + 12], (n, 8))
marktools.Articulation('>')(chord)
# make the upper voice
v_higher = Voice([chord])
v_higher.name = 'rh_higher'
marktools.LilyPondCommandMark('voiceOne')(v_higher)
# combine them together
p = Container([v_lower, v_higher])
p.is_parallel = True
# make all 1/8 beats breakable
for n in v_lower.leaves[:-1]:
    marktools.BarLine('')(n)
return p

```

Now we can call this function to create any number of *cells*. That was actually the hardest part of reconstructing the opening of Ligeti's *Désordre*. Because the repetition of patterns occurs also at the level of measures and staves, we will now define functions to create these other higher level constructs.

2.3.2 The measure

We define a function to create a measure from a list of lists of numbers:

```

def measure_build(pitches):
    '''Constructs a measure composed of *Désordre cells*.
    'pitches' is a list of lists of number (e.g., [[1, 2, 3], [2, 3, 4]])
    The function returns a DynamicMeasure.
    '''
    result = measuretools.DynamicMeasure([ ])
    for seq in pitches:
        result.append(desordre_cell(seq))
    # make denominator 8
    if contexttools.get_effective_time_signature(result).denominator == 1:
        result.denominator = 8
    return result

```

The function is very simple. It simply creates a `DynamicMeasure` and then populates it with *cells* that are created internally with the function previously defined. The function takes a list *pitches* which is actually a list of lists of pitches (e.g., `[[1, 2, 3], [2, 3, 4]]`). The list of lists of pitches is iterated to create each of the *cells* to be appended to the `DynamicMeasures`. We could have defined the function to take ready made *cells* directly, but we are building the hierarchy of functions so that we can pass simple lists of lists of numbers to generate the full structure. To construct a Ligeti measure we would call the function like so:

```

>>> measure = measure_build([[0, 4, 7], [0, 4, 7, 9], [4, 7, 9, 11]])
>>> show(Staff([measure]))

```



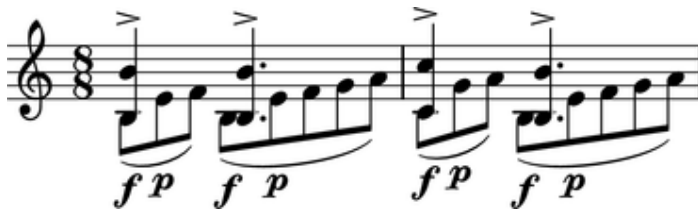
2.3.3 The staff

Now we move up to the next level, the staff:

```
def staff_build(pitches):
    '''Returns a Staff containing DynamicMeasures.'''
    result = Staff([])
    for seq in pitches:
        measure = measure_build(seq)
        result.append(measure)
    return result
```

The function again takes a plain list as argument. The list must be a list of lists (for measures) of lists (for cells) of pitches. The function simply constructs the Ligeti measures internally by calling our previously defined function and puts them inside a Staff. As with measures, we can now create full measure sequences with this new function:

```
>>> pitches = [[[-1, 4, 5], [-1, 4, 5, 7, 9]], [[0, 7, 9], [-1, 4, 5, 7, 9]]]
>>> staff = staff_build(pitches)
>>> show(staff)
```



2.3.4 The score

Finally a function that will generate the whole opening section of the piece *Désordre*:

```
def desordre_build(pitches):
    '''Returns a complete PianoStaff with Ligeti music!'''
    assert len(pitches) == 2
    piano = scoretools.PianoStaff([])
    # build the music...
    for hand in pitches:
        seq = staff_build(hand)
        piano.append(seq)
    # set clef and key signature to left hand staff...
    contexttools.ClefMark('bass')(piano[1])
    contexttools.KeySignatureMark('b', 'major')(piano[1])
    return piano
```

The function creates a PianoStaff, constructs Staves with Ligeti music and appends these to the empty PianoStaff. Finally it sets the clef and key signature of the lower staff to match the original score. The argument of the function is a list of length 2, depth 3. The first element in the list corresponds to the upper staff, the second to the lower staff.

The final result:

```
>>> top = [[[-1, 4, 5], [-1, 4, 5, 7, 9]], [[0, 7, 9], [-1, 4, 5, 7, 9]], [[2, 4, 5, 7, 9], [0, 5, 7, 9]], [[-9, -4, -2], [-9, -4, -2, 1, 3]], [[-6, -2, 1], [-9, -4, -2, 1, 3]], [[-4, -2, 1, 3, 5], [-4, -2, 1, 3, 5]]]
>>> bottom = [[[-9, -4, -2], [-9, -4, -2, 1, 3]], [[-6, -2, 1], [-9, -4, -2, 1, 3]], [[-4, -2, 1, 3, 5], [-4, -2, 1, 3, 5]]]

>>> desordre = desordre_build([top, bottom])

>>> from abjad.tools import documentationtools
>>> lilypond_file = documentationtools.make_ligeti_example_lilypond_file(desordre)

>>> show(lilypond_file)
```



Now that we have the redundant aspect of the piece compactly expressed and encapsulated, we can play around with it by changing the sequence of pitches.

In order for each staff to carry its own sequence of independent measure changes, LilyPond requires some special setting up prior to rendering. Specifically, one must move the `LilyPond Timing_translator` out from the score context and into the staff context.

(You can refer to the LilyPond documentation on [Polymetric notation](#) to learn all about how this works.)

In this example we use a custom `documentationtools` function to set up our LilyPond file automatically.

2.4 Mozart: *Musikalisches Würfelspiel*

Mozart's dice game is a method for aleatorically generating sixteen-measure-long minuets. For each measure, two six-sided dice are rolled, and the sum of the dice used to look up a measure number in one of two tables (one for each half of the minuet). The measure number then locates a single measure from a collection of musical fragments. The fragments are concatenated together, and "music" results.

Implementing the dice game in a composition environment is somewhat akin to (although also somewhat more complicated than) the ubiquitous `hello world program` which every programming language uses to demonstrate its basic syntax.

Note: The musical dice game in question (*k516f*) has long been attributed to Mozart, albeit inconclusively. Its actual provenance is a musicological problem with which we are unconcerned here.

2.4.1 The materials

At the heart of the dice game is a large collection, *or corpus*, of musical fragments. Each fragment is a single 3/8 measure, consisting of a treble voice and a bass voice. Traditionally, these fragments are stored in a “score”, or “table of measures”, and located via two tables of measure numbers, which act as lookups, indexing into that collection.

Duplicate measures in the original corpus are common. Notably, the 8th measure - actually a pair of measures represent the first and second alternate ending of the first half of the minuet - are always identical. The last measure of the piece is similarly limited - there are only two possibilities rather than the usual eleven (for the numbers 2 to 12, being all the possible sums of two 6-sided dice).

How might we store this corpus compactly?

Some basic musical information in Abjad can be stored as strings, rather than actual collections of class instances. Abjad can parse simple LilyPond strings via `p`, which interprets a subset of LilyPond syntax, and understands basic concepts like notes, chords, rests and skips, as well as beams, slurs, ties, and articulations.

```
>>> lily_string = r"\new Staff { c'4 ( d'4 <cs' e'>8 ) -. r8 <g' b' d''>4 ^ \marcato ~ <g' b' d''>1 }"
>>> parsed_result = p(lily_string)
>>> f(parsed_result)
\new Staff {
  c'4 (
    d'4
    <cs' e'>8 -\staccato )
  r8
  <g' b' d''>4 ^\marcato ~
  <g' b' d''>1
}
>>> show(parsed_result)
```



So, instead of storing our musical information as Abjad components, we’ll represent each fragment in the corpus as a pair of strings: one representing the bass voice contents, and the other representing the treble. This pair of strings can be packaged together into a collection. For this implementation, we’ll package them into a dictionary. Python dictionaries are cheap, and often provide more clarity than lists; the composer does not have to rely on remembering a convention for what data should appear in which position in a list - they can simply label that data semantically. In our musical dictionary, the treble voice will use the key ‘t’ and the bass voice will use the key ‘b’.

```
>>> fragment = {'t': "g''8 ( e''8 c''8 )", 'b': '<c e>4 r8'}
```

Instead of relying on measure number tables to find our fragments - as in the original implementation, we’ll package our fragment dictionaries into a list of lists of fragment dictionaries. That is to say, each of the sixteen measures in the piece will be represented by a list of fragment dictionaries. Furthermore, the 8th measure, which breaks the pattern, will simply be a list of two fragment dictionaries. Structuring our information in this way lets us avoid using measure

WOLFGANG AMADEUS MOZART

Musikalisches Würfelspiel

Table of Measure Numbers

Part One

	I	II	III	IV	V	VI	VII	VIII
2	96	22	141	41	105	122	11	30
3	32	6	128	63	146	46	134	81
4	69	95	158	13	153	55	110	24
5	40	17	113	85	161	2	159	100
6	148	74	163	45	80	97	36	107
7	104	157	27	167	154	68	118	91
8	152	60	171	53	99	133	21	127
9	119	84	114	50	140	86	169	94
10	98	142	42	156	75	129	62	123
11	3	87	165	61	135	47	147	33
12	54	130	10	103	28	37	106	5

Part Two

	I	II	III	IV	V	VI	VII	VIII
2	70	121	26	9	112	49	109	14
3	117	39	126	56	174	18	116	83
4	66	139	15	132	73	58	145	79
5	90	176	7	34	67	160	52	170
6	25	143	64	125	76	136	1	93
7	138	71	150	29	101	162	23	151
8	16	155	57	175	43	168	89	172
9	120	88	48	166	51	115	72	111
10	65	77	19	82	137	38	149	8
11	102	4	31	164	144	59	173	78
12	35	20	108	92	12	124	44	131

Table of Measures



Figure 2.1: Part of a pen-and-paper implementation from the 20th century.

number tables entirely; Python's list-indexing affordances will take care of that for us. The complete corpus looks like this:

```
mozart_measures = [
    [ # measure 1 choices
      {'b': 'c4 r8', 't': "e''8 c''8 g'8"},
      {'b': '<c e>4 r8', 't': "g'8 c''8 e''8"},
      {'b': '<c e>4 r8', 't': "g''8 ( e''8 c''8 )"},
      {'b': '<c e>4 r8', 't': "c''16 b'16 c''16 e''16 g'16 c''16"},
      {'b': '<c e>4 r8', 't': "c'''16 b'16 c'''16 g''16 e''16 c''16"},
      {'b': 'c4 r8', 't': "e''16 d''16 e''16 g''16 c'''16 g'16"},
      {'b': '<c e>4 r8', 't': "g''8 f''16 e''16 d''16 c''16"},
      {'b': '<c e>4 r8', 't': "e''16 c''16 g''16 e''16 c'''16 g''16"},
      {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "c''8 g'8 e''8"},
      {'b': '<c e>4 r8', 't': "g''8 c''8 e''8"},
      {'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"},
    ],
    [ # measure 2 choices
      {'b': 'c4 r8', 't': "e''8 c''8 g'8"},
      {'b': '<c e>4 r8', 't': "g'8 c''8 e''8"},
      {'b': '<c e>4 r8', 't': "g''8 e''8 c''8"},
      {'b': '<e g>4 r8', 't': "c''16 g'16 c''16 e''16 g'16 c''16"},
      {'b': '<c e>4 r8', 't': "c'''16 b'16 c'''16 g''16 e''16 c''16"},
      {'b': 'c4 r8', 't': "e''16 d''16 e''16 g''16 c'''16 g'16"},
      {'b': '<c e>4 r8', 't': "g''8 f''16 e''16 d''16 c''16"},
      {'b': '<c e>4 r8', 't': "c''16 g'16 e''16 c''16 g'16 e''16"},
      {'b': '<c e>4 r8', 't': "c''8 g'8 e''8"},
      {'b': '<c e>4 <c g>8', 't': "g''8 c''8 e''8"},
      {'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"},
    ],
    [ # measure 3 choices
      {'b': '<b, g>4 g,8', 't': "d''16 e''16 f''16 d''16 c''16 b'16"},
      {'b': 'g,4 r8', 't': "b'8 d''8 g''8"},
      {'b': 'g,4 r8', 't': "b'8 d''16 b'16 a'16 g'16"},
      {'b': '<g b>4 r8', 't': "f''8 d''8 b'8"},
      {'b': '<b, d>4 r8', 't': "g''16 fs''16 g''16 d''16 b'16 g'16"},
      {'b': '<g b>4 r8', 't': "f''16 e''16 f''16 d''16 c''16 b'16"},
      {'b': '<g, g>4 <b, g>8', 't': "b'16 c''16 d''16 e''16 f''16 d''16"},
      {'b': 'g8 g8 g8', 't': "<b' d''>8 <b' d''>8 <b' d''>8"},
      {'b': 'g,4 r8', 't': "b'16 c''16 d''16 b'16 a'16 g'16"},
      {'b': 'b,4 r8', 't': "d''8 ( b'8 g'8 )"},
      {'b': 'g4 r8', 't': "b'16 a'16 b'16 c''16 d''16 b'16"},
    ],
    [ # measure 4 choices
      {'b': '<c e>4 r8', 't': "c''16 b'16 c''16 e''16 g'8"},
      {'b': 'c4 r8', 't': "e''16 c''16 b'16 c''16 g'8"},
      {'b': '<e g>4 r8', 't': "c''8 ( g'8 e'8 )"},
      {'b': '<e g>4 r8', 't': "c''8 e''8 g'8"},
      {'b': '<e g>4 r8', 't': "c''16 b'16 c''16 g'16 e'16 c'16"},
      {'b': '<c e>4 r8', 't': "c''8 c''16 d''16 e''8"},
      {'b': 'c4 r8', 't': "<c'' e''>8 <c'' e''>16 <d'' f''>16 <e'' g''>8"},
      {'b': '<e g>4 r8', 't': "c''8 e''16 c''16 g'8"},
      {'b': '<e g>4 r8', 't': "c''16 g'16 e''16 c''16 g''8"},
      {'b': '<e g>4 r8', 't': "c''8 e''16 c''16 g''8"},
      {'b': '<e g>4 r8', 't': "c''16 e''16 c''16 g'16 e'8"},
    ],
    [ # measure 5 choices
      {'b': 'c4 r8', 't': "fs''8 a''16 fs''16 d''16 fs''16"},
      {'b': 'c8 c8 c8', 't': "<fs' d''>8 <d'' fs''>8 <fs'' a''>8"},
    ]
]
```

```

{'b': 'c4 r8', 't': "d''16 a'16 fs''16 d''16 a''16 fs''16"},
{'b': 'c8 c8 c8', 't': "<fs' d''>8 <fs' d''>8 <fs' d''>8"},
{'b': 'c4 r8', 't': "d''8 a'8 ^\\turn fs''8"},
{'b': 'c4 r8', 't': "d''16 cs''16 d''16 fs''16 a''16 fs''16"},
{'b': '<c a>4 <c a>8', 't': "fs''8 a''8 d''8"},
{'b': '<c fs>8 <c fs>8 <c a>8', 't': "a'8 a'16 d''16 fs''8"},
{'b': 'c8 c8 c8', 't': "<d'' fs''>8 <d'' fs''>8 <d'' fs''>8"},
{'b': '<c d>8 <c d>8 <c d>8', 't': "fs''8 fs''16 d''16 a''8"},
{'b': '<c a>4 r8', 't': "fs''16 d''16 a'16 a''16 fs''16 d''16"},
],
[ # measure 6 choices
{'b': '<b, d>8 <b, d>8 <b, d>8', 't': "g''16 fs''16 g''16 b''16 d''8"},
{'b': '<b, d>4 r8', 't': "g''8 b''16 g''16 d''16 b'16"},
{'b': '<b, d>4 r8', 't': "g''8 b''8 d''8"},
{'b': '<b, g>4 r8', 't': "a'8 fs'16 g'16 b'16 g''16"},
{'b': '<b, d>4 <b, g>8', 't': "g''16 fs''16 g''16 d''16 b'16 g'16"},
{'b': 'b,4 r8', 't': "g''8 b''16 g''16 d''16 g''16"},
{'b': '<b, g>4 r8', 't': "d''8 g''16 d''16 b'16 d''16"},
{'b': '<b, g>4 r8', 't': "d''8 d''16 g''16 b''8"},
{'b': '<b, d>8 <b, d>8 <b, g>8', 't': "a''16 g''16 fs''16 g''16 d''8"},
{'b': '<b, d>4 r8', 't': "g''8 g''16 d''16 b''8"},
{'b': '<b, d>4 r8', 't': "g''16 b''16 g''16 d''16 b'8"},
],
[ # measure 7 choices
{'b': 'c8 d8 d,8', 't': "e''16 c''16 b'16 a'16 g'16 fs'16"},
{'b': 'c8 d8 d,8', 't': "a'16 e''16 <b' d''>16 <a' c''>16 <g' b'>16 <fs' a'>16"},
{'b': 'c8 d8 d,8', 't': "<b' d''>16 ( <a' c''>16 ) <a' c''>16 ( <g' b'>16 ) <g' b'>16 ( <fs' a'>16 )"},
{'b': 'c8 d8 d,8', 't': "e''16 g''16 d''16 c''16 b'16 a'16"},
{'b': 'c8 d8 d,8', 't': "a'16 e''16 d''16 g''16 fs''16 a''16"},
{'b': 'c8 d8 d,8', 't': "e''16 a''16 g''16 b''16 fs''16 a''16"},
{'b': 'c8 d8 d,8', 't': "c''16 e''16 g''16 d''16 a'16 fs''16"},
{'b': 'c8 d8 d,8', 't': "e''16 g''16 d''16 g''16 a'16 fs''16"},
{'b': 'c8 d8 d,8', 't': "e''16 c''16 b'16 g'16 a'16 fs'16"},
{'b': 'c8 d8 d,8', 't': "e''16 c''16 b''16 g''16 a'16 fs''16"},
{'b': 'c8 d8 d,8', 't': "a'8 d''16 c''16 b'16 a'16"},
],
[ # measure 8 choices (always using both)
{'b': 'g,8 g16 f16 e16 d16', 't': "<g' b' d'' g''>4 r8"},
{'b': 'g,8 b16 g16 fs16 e16', 't': "<g' b' d'' g''>4 r8"},
],
[ # measure 9 choices
{'b': 'd4 c8', 't': "fs''8 a''16 fs''16 d''16 fs''16"},
{'b': '<d fs>4 r8', 't': "d''16 a'16 d''16 fs''16 a''16 fs''16"},
{'b': '<d a>8 <d fs>8 <c d>8', 't': "fs''8 a''8 fs''8"},
{'b': '<c a>4 <c a>8', 't': "fs''16 a''16 d''16 a''16 fs''16 a''16"},
{'b': 'd4 c8', 't': "d'16 fs'16 a'16 d''16 fs'16 a'16"},
{'b': 'd,16 d16 cs16 d16 c16 d16', 't': "<a' d'' fs''>8 fs''4 ^\\tr"},
{'b': '<d fs>4 <c fs>8', 't': "a''8 ( fs''8 d''8 )"},
{'b': '<d fs>4 <c fs>8', 't': "d''8 a''16 fs''16 d''16 a'16"},
{'b': '<d fs>4 r8', 't': "d''16 a'16 d''8 fs''8"},
{'b': '<c a>4 <c a>8', 't': "fs''16 d''16 a'8 fs''8"},
{'b': '<d fs>4 <c a>8', 't': "a'8 d''8 fs''8"},
],
[ # measure 10 choices
{'b': '<b, g>4 r8', 't': "g''8 b''16 g''16 d''8"},
{'b': 'b,16 d16 g16 d16 b,16 g,16', 't': "g''8 g'8 g'8"},
{'b': 'b,4 r8', 't': "g''16 b''16 g''16 b''16 d''8"},
{'b': '<b, d>4 <b, d>8', 't': "a''16 g''16 b''16 g''16 d''16 g''16"},

```

```

{'b': '<b, d>4 <b, d>8', 't': "g''8 d''16 b'16 g'8"},
{'b': '<b, d>4 <b, d>8', 't': "g''16 b''16 d'''16 b''16 g''8"},
{'b': '<b, d>4 r8', 't': "g''16 b''16 g''16 d''16 b'16 g'16"},
{'b': '<b, d>4 <b, d>8', 't': "g''16 d''16 g''16 b''16 g''16 d''16"},
{'b': '<b, d>4 <b, g>8', 't': "g''16 b''16 g''8 d''8"},
{'b': 'g,16 b,16 g8 b,8', 't': "g''8 d''4 ^\\tr"},
{'b': 'b,4 r8', 't': "g''8 b''16 d'''16 d''8"},
],
[ # measure 11 choices
{'b': 'c16 e16 g16 e16 c'16 c16', 't': "<c'' e''>8 <c'' e''>8 <c'' e''>8"},
{'b': 'e4 e16 c16', 't': "c''16 g'16 c''16 e''16 g''16 <c'' e''>16"},
{'b': '<c g>4 <c e>8', 't': "e''8 g''16 e''16 c''8"},
{'b': '<c g>4 r8', 't': "e''16 c''16 e''16 g''16 c'''16 g''16"},
{'b': '<c g>4 <c g>8', 't': "e''16 g''16 c''16 g''16 e''16 c'16"},
{'b': 'c16 b,16 c16 d16 e16 fs16', 't': "<g' c'' e''>8 e''4 ^\\tr"},
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "e''8 c''8 g'8"},
{'b': '<c g>4 <c e>8', 't': "e''8 c''16 e''16 g''16 c'''16"},
{'b': '<c g>4 <c e>8', 't': "e''16 c''16 e''8 g''8"},
{'b': '<c g>4 <c g>8', 't': "e''16 c''16 g'8 e''8"},
{'b': '<c g>4 <c e>8', 't': "e''8 ( g''8 c''8 )"},
],
[ # measure 12 choices
{'b': 'g4 g,8', 't': "<c'' e''>8 <b' d''>8 r8"},
{'b': '<g, g>4 g8', 't': "d''16 b'16 g'8 r8"},
{'b': 'g8 g,8 r8', 't': "<c'' e''>8 <b' d''>16 <g' b'>16 g'8"},
{'b': 'g4 r8', 't': "e''16 c''16 d''16 b'16 g'8"},
{'b': 'g8 g,8 r8', 't': "g''16 e''16 d''16 b'16 g'8"},
{'b': 'g4 g,8', 't': "b'16 d''16 g''16 d''16 b'8"},
{'b': 'g8 g,8 r8', 't': "e''16 c''16 b'16 d''16 g''8"},
{'b': '<g b>4 r8', 't': "d''16 b''16 g''16 d''16 b'8"},
{'b': '<b, g>4 <b, d>8', 't': "d''16 b'16 g'8 g''8"},
{'b': 'g16 fs16 g16 d16 b,16 g,16', 't': "d''8 g'4"},
],
[ # measure 13 choices
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "e''8 c''8 g'8"},
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g'8 c''8 e''8"},
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g''8 e''8 c''8"},
{'b': '<c e>4 <e g>8', 't': "c''16 b'16 c''16 e''16 g'16 c''16"},
{'b': '<c e>4 <c g>8', 't': "c'''16 b''16 c'''16 g''16 e''16 c''16"},
{'b': '<c g>4 <c e>8', 't': "e''16 d''16 e''16 g''16 c'''16 g''16"},
{'b': '<c e>4 r8', 't': "g''8 f''16 e''16 d''16 c''16"},
{'b': '<c e>4 r8', 't': "c''16 g'16 e''16 c''16 g''16 e''16"},
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "c''8 g'8 e''8"},
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g''8 c''8 e''8"},
{'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"},
],
[ # measure 14 choices
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "e''8 ( c''8 g'8 )"},
{'b': '<c e>4 <c g>8', 't': "g'8 ( c''8 e''8 )"},
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g''8 e''8 c''8"},
{'b': '<c e>4 <c e>8', 't': "c''16 b'16 c''16 e''16 g'16 c''16"},
{'b': '<c e>4 r8', 't': "c'''16 b''16 c'''16 g''16 e''16 c''16"},
{'b': '<c g>4 <c e>8', 't': "e''16 d''16 e''16 g''16 c'''16 g''16"},
{'b': '<c e>4 <e g>8', 't': "g''8 f''16 e''16 d''16 c''16"},
{'b': '<c e>4 r8', 't': "c''16 g'16 e''16 c''16 g''16 e''16"},
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "c''8 g'8 e''8"},
{'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g''8 c''8 e''8"},
{'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"},
],

```

```

],
[ # measure 15 choices
  {'b': "<f a>4 <g d'>8", 't': "d''16 f''16 d''16 f''16 b'16 d''16"},
  {'b': 'f4 g8', 't': "d''16 f''16 a''16 f''16 d''16 b'16"},
  {'b': 'f4 g8', 't': "d''16 f''16 a'16 d''16 b'16 d''16"},
  {'b': 'f4 g8', 't': "d''16 ( cs''16 ) d''16 f''16 g'16 b'16"},
  {'b': 'f8 d8 g8', 't': "f''8 d''8 g''8"},
  {'b': 'f16 e16 d16 e16 f16 g16', 't': "f''16 e''16 d''16 e''16 f''16 g''16"},
  {'b': 'f16 e16 d8 g8', 't': "f''16 e''16 d''8 g''8"},
  {'b': 'f4 g8', 't': "f''16 e''16 d''16 c''16 b'16 d''16"},
  {'b': 'f4 g8', 't': "f''16 d''16 a'8 b'8"},
  {'b': 'f4 g8', 't': "f''16 a''16 a'8 b'16 d''16"},
  {'b': 'f4 g8', 't': "a'8 f''16 d''16 a'16 b'16"},
],
[ # measure 16 choices
  {'b': 'c8 g,8 c,8', 't': "c''4 r8"},
  {'b': 'c4 c,8', 't': "c''8 c'8 r8"},
],
]

```

We can then use the `p()` function we saw earlier to “build” the treble and bass components of a measure like this:

```

def build_one_mozart_measure(measure_dict):
    # parse the contents of a measure definition dictionary
    # wrap the expression to be parsed inside a LilyPond { } block
    treble = p('{{ {} }}'.format(measure_dict['t']))
    bass = p('{{ {} }}'.format(measure_dict['b']))
    return treble, bass

```

Let’s try with a measure-definition of our own:

```

>>> my_measure_dict = {'b': 'c4 ^\\tr r8', 't': "e''8 ( c''8 g'8 )"}
>>> treble, bass = build_one_mozart_measure(my_measure_dict)

```

Traceback (most recent call last):

```

File "<stdin>", line 1, in <module>
File "<stdin>", line 5, in build_one_mozart_measure
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/iotools/p.py", line 33, in p
    return LilyPondParser()(args[0])
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/lilypondparsertools/LilyPondParser/LilyPondParser.py", line 10, in __init__
    lexer=self._lexer)
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/lilypondparsertools/_parse.py", line 66, in _parse
    self.lookahead = get_token() # Get the next token
File "/usr/local/lib/python2.7/dist-packages/ply/lex.py", line 348, in token
    newtok = func(tok)
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/lilypondparsertools/_LilyPondLexicalDefinition.py", line 10, in scan
    t.type = self.scan_escaped_word(t)
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/lilypondparsertools/_LilyPondLexicalDefinition.py", line 10, in scan_escaped_word
    raise Exception('Unknown escaped word "%s".' % t.value)

```

Exception: Unknown escaped word "\tr".

```

>>> f(treble)

```

Traceback (most recent call last):

```

File "<stdin>", line 1, in <module>
NameError: name 'treble' is not defined

```

```

>>> f(bass)

```

Traceback (most recent call last):

```

File "<stdin>", line 1, in <module>
NameError: name 'bass' is not defined

```

Now with one from the Mozart measure collection defined earlier. We’ll grab the very last choice for the very last

measure:

```
>>> my_measure_dict = mozart_measures[-1][-1]
>>> treble, bass = build_one_mozart_measure(my_measure_dict)
>>> f(treble)
{
    c''8
    c'8
    r8
}
>>> f(bass)
{
    c4
    c,8
}
```

2.4.2 The structure

After storing all of the musical fragments into a corpus, concatenating those elements into a musical structure is relatively trivial. We'll use the `choice()` function from Python's `random` module. `random.choice()` randomly selects one element from an input list.

```
>>> import random
>>> my_list = [1, 'b', 3]
>>> my_result = [random.choice(my_list) for i in range(20)]
>>> my_result
['b', 'b', 1, 3, 3, 1, 'b', 3, 1, 'b', 'b', 3, 'b', 'b', 1, 'b', 'b', 1, 3, 3]
```

Our corpus is a list comprising sixteen sublists, one for each measure in the minuet. To build our musical structure, we can simply iterate through the corpus and call `choice` on each sublist, appending the chosen results to another list. The only catch is that the *eighth* measure of our minuet is actually the first-and-second-ending for the repeat of the first phrase. The sublist of the corpus for measure eight contains *only* the first and second ending definitions, and both of those measures should appear in the final piece, always in the same order. We'll have to intercept that sublist while we iterate through the corpus and apply some different logic.

The easiest way to intercept measure eight is to use the Python builtin `enumerate`, which allows you to iterate through a collection while also getting the index of each element in that collection:

```
def choose_mozart_measures():
    chosen_measures = []
    for i, choices in enumerate(mozart_measures):
        if i == 7: # get both alternative endings for mm. 8
            chosen_measures.extend(choices)
        else:
            choice = random.choice(choices)
            chosen_measures.append(choice)
    return chosen_measures
```

Note: In `choose_mozart_measures` we test for index 7, rather than 8, because list indices count from 0 instead of 1.

The result will be a *seventeen*-item-long list of measure definitions:

```
>>> choices = choose_mozart_measures()
>>> for i, measure in enumerate(choices):
...     print i, measure
...
```

```
0 {'b': 'c4 r8', 't': "e''8 c''8 g'8"}
1 {'b': '<c e>4 r8', 't': "c''8 g'8 e''8"}
2 {'b': 'g,4 r8', 't': "b'8 d''8 g''8"}
3 {'b': '<e g>4 r8', 't': "c''16 e''16 c''16 g'16 e'8"}
4 {'b': '<c fs>8 <c fs>8 <c a>8', 't': "a'8 a'16 d''16 fs''8"}
5 {'b': '<b, g>4 r8', 't': "a'8 fs'16 g'16 b'16 g''16"}
6 {'b': 'c8 d8 d,8', 't': "e''16 c''16 b''16 g''16 a''16 fs''16"}
7 {'b': 'g,8 g16 f16 e16 d16', 't': "<g' b' d'' g''>4 r8"}
8 {'b': 'g,8 b16 g16 fs16 e16', 't': "<g' b' d'' g''>4 r8"}
9 {'b': '<c a>4 <c a>8', 't': "fs''16 d''16 a'8 fs''8"}
10 {'b': '<b, d>4 r8', 't': "g''16 b''16 g''16 d''16 b'16 g'16"}
11 {'b': '<c g>4 <c e>8', 't': "e''8 g''16 e''16 c''8"}
12 {'b': 'g8 g,8 r8', 't': "<c'' e''>8 <b' d''>16 <g' b'>16 g'8"}
13 {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "c''8 g'8 e''8"}
14 {'b': '<c e>4 <c g>8', 't': "g'8 ( c''8 e''8 )"}
15 {'b': 'f4 g8', 't': "d''16 f''16 a'16 d''16 b'16 d''16"}
16 {'b': 'c8 g,8 c,8', 't': "c''4 r8"}
```

2.4.3 The score

Now that we have our raw materials, and a way to organize them, we can start building our score. The tricky part here is figuring out how to implement LilyPond’s repeat structure in Abjad. LilyPond structures its repeats something like this:

```
\repeat volta n {
    music to be repeated
}

\alternative {
    { ending 1 }
    { ending 2 }
    { ending n }
}

...music after the repeat...
```

What you see above is really just two containers, each with a little text (“repeat volta n” and “alternative”) prepended to their opening curly brace. To create that structure in Abjad, we’ll need to use the `LilyPondCommandMark` class, which allows you to place LilyPond commands like “break” relative to any score component:

```
>>> con = Container("c'4 d'4 e'4 f'4")
>>> mark = marktools.LilyPondCommandMark('before-the-container', 'before')(con)
>>> mark = marktools.LilyPondCommandMark('after-the-container', 'after')(con)
>>> mark = marktools.LilyPondCommandMark('opening-of-the-container', 'opening')(con)
>>> mark = marktools.LilyPondCommandMark('closing-of-the-container', 'closing')(con)
>>> mark = marktools.LilyPondCommandMark('to-the-right-of-a-note', 'right')(con[2])
>>> f(con)
\before-the-container
{
    \opening-of-the-container
    c'4
    d'4
    e'4 \to-the-right-of-a-note
    f'4
    \closing-of-the-container
}
\after-the-container
```

Notice the second argument to each `LilyPondCommandMark` above, like *before* and *closing*. These are format slot indications, which control where the command is placed in the LilyPond code relative to the score element it is attached to. To mimic LilyPond’s repeat syntax, we’ll have to create two `LilyPondCommandMark` instances, both using the “before” format slot, insuring that their command is placed before their container’s opening curly brace.

Now let’s take a look at the code that puts our score together:

```
def build_mozart_piano_staff():
    treble_staff = Staff([])
    bass_staff = Staff([])
    # select the measures to use
    choices = choose_mozart_measures()
    # create and populate the volta containers
    treble_volta = Container([])
    bass_volta = Container([])
    for choice in choices[:7]:
        treble, bass = build_one_mozart_measure(choice)
        treble_volta.append(treble)
        bass_volta.append(bass)
    # add marks to the volta containers
    marktools.LilyPondCommandMark('repeat volta 2', 'before')(treble_volta)
    marktools.LilyPondCommandMark('repeat volta 2', 'before')(bass_volta)
    # add the volta containers to our staves
    treble_staff.append(treble_volta)
    bass_staff.append(bass_volta)
    # create and populate the alternative ending containers
    treble_alternative = Container([])
    bass_alternative = Container([])
    for choice in choices[7:9]:
        treble, bass = build_one_mozart_measure(choice)
        treble_alternative.append(treble)
        bass_alternative.append(bass)
    # add marks to the alternative containers
    marktools.LilyPondCommandMark('alternative', 'before')(treble_alternative)
    marktools.LilyPondCommandMark('alternative', 'before')(bass_alternative)
    # add the alternative containers to our staves
    treble_staff.append(treble_alternative)
    bass_staff.append(bass_alternative)
    # create the remaining measures
    for choice in choices[9:]:
        treble, bass = build_one_mozart_measure(choice)
        treble_staff.append(treble)
        bass_staff.append(bass)
    # add meter
    contexttools.TimeSignatureMark((3, 8))(treble_staff)
    # add bass clef
    contexttools.ClefMark('bass')(bass_staff)
    # add the final double bar line at the end of each final measure
    marktools.BarLine('|.') (treble_staff[-1])
    marktools.BarLine('|.') (bass_staff[-1])
    # combine into a PianoStaff
    piano_staff = scoretools.PianoStaff([treble_staff, bass_staff])
    # add an instrument name via contexttools.InstrumentMark
    contexttools.InstrumentMark('Katzenklavier', 'kk.',
        target_context = scoretools.PianoStaff)(piano_staff)
    return piano_staff

>>> piano_staff = build_mozart_piano_staff()
Traceback (most recent call last):
```

```

File "<stdin>", line 1, in <module>
File "<stdin>", line 34, in build_mozart_piano_staff
File "<stdin>", line 4, in build_one_mozart_measure
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/iotools/p.py", line 33, in p
    return LilyPondParser()(args[0])
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/lilypondparsertools/LilyPondParser/LilyPondParser.py", line 10, in LilyPondParser
    lexer=self._lexer)
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/lilypondparsertools/_parse.py", line 66, in _parse
    self.lookahead = get_token() # Get the next token
File "/usr/local/lib/python2.7/dist-packages/ply/lex.py", line 348, in token
    newtok = func(tok)
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/lilypondparsertools/_LilyPondLexicalDefinition.py", line 10, in _LilyPondLexicalDefinition
    t.type = self.scan_escaped_word(t)
File "/media/Work/dev/scores/abjad/trunk/abjad/tools/lilypondparsertools/_LilyPondLexicalDefinition.py", line 10, in _LilyPondLexicalDefinition
    raise Exception('Unknown escaped word "%s".' % t.value)
Exception: Unknown escaped word "\tr".
>>> show(piano_staff)

```

The image displays three staves of musical notation. The first staff, labeled 'atzenklavier', shows a sequence of notes and rests in 3/8 time. The second staff, labeled 'Kk.', includes a first ending bracket and a repeat sign. The third staff, also labeled 'Kk.', continues the musical sequence. The notation is complex, with many beamed notes and various accidentals.

```

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'piano_staff' is not defined

```

Note: Our instrument name got cut off! Looks like we need to do a little formatting. Keep reading...

2.4.4 The document

As you can see above, we've now got our randomized minuet. However, we can still go a bit further. LilyPond provides a wide variety of settings for controlling the overall *look* of a musical document, often through its *header*, *layout* and *paper* blocks. Abjad, in turn, gives us object-oriented access to these settings through the its *lilypondfiletools* module.

We'll use `abjad.tools.lilypondfiletools.make_basic_lilypond_file()` to wrap our `PianoStaff` inside a `LilyPondFile` instance. From there we can access the other "blocks" of our document to add a title, a composer's name, change the global staff size, paper size, staff spacing and so forth.


```
def build_mozart_lily(piano_staff):
    # wrap the PianoStaff with a LilyPondFile
    lily = lilypondfiletools.make_basic_lilypond_file(piano_staff)
    # create some markup to use in our header block
    title = markuptools.Markup('\bold \sans "Ein Musikalisches Wuerfelspiel"')
    composer = schemetools.Scheme("W. A. Mozart (maybe?)")
    # change various settings
    lily.global_staff_size = 12
    lily.header_block.title = title
    lily.header_block.composer = composer
    lily.layout_block.ragged_right = True
    lily.paper_block.markup_system_spacing__basic_distance = 8
    lily.paper_block.paper_width = 180
    return lily
```

```
>>> lily = build_mozart_lily(piano_staff)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'piano_staff' is not defined
>>> print lily
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'lily' is not defined
```

```
>>> print lily.header_block
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'lily' is not defined
>>> f(lily.header_block)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'lily' is not defined
```

```
>>> print lily.layout_block
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'lily' is not defined
>>> f(lily.layout_block)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'lily' is not defined
```

```
>>> print lily.paper_block
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'lily' is not defined
>>> f(lily.paper_block)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'lily' is not defined
```

And now the final result:

```
>>> show(lily)
```

Ein Musikalisches Wuerfelspiel

W. A. Mozart (maybe?)



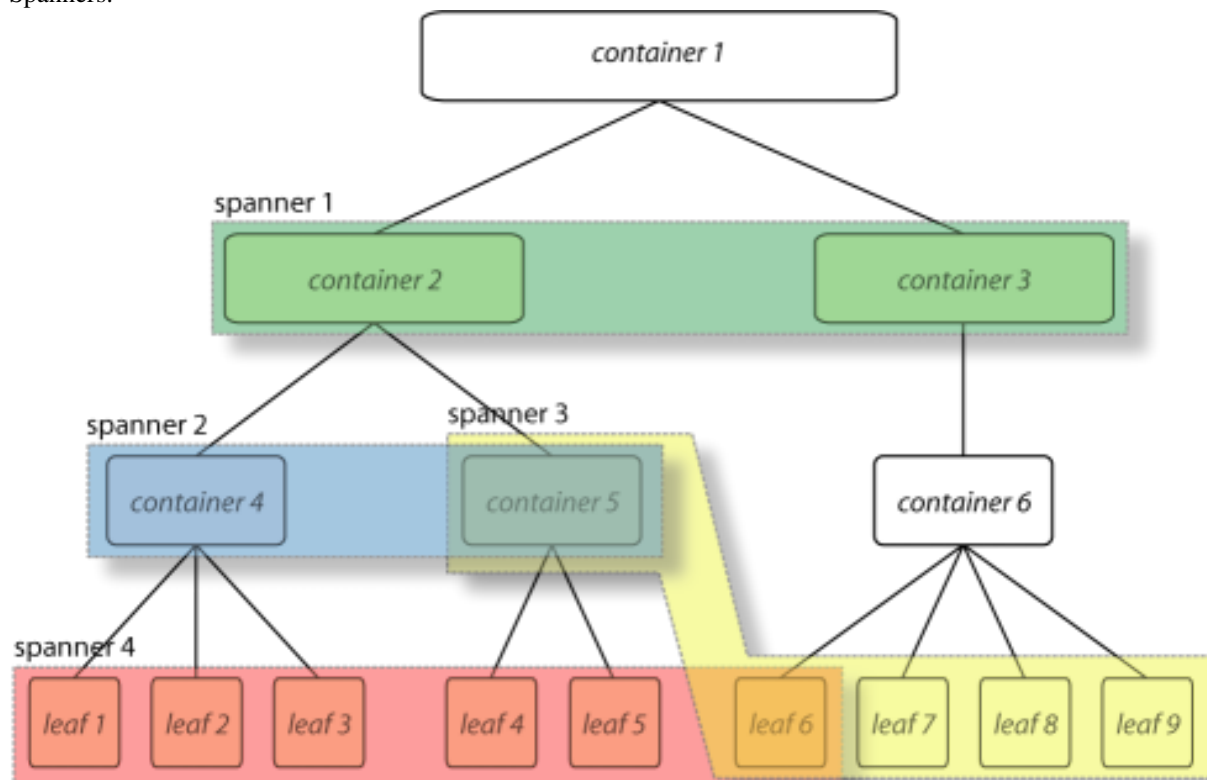
```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'lily' is not defined
```

SYSTEM OVERVIEW

3.1 Leaf, Container, Spanner, Mark

At the heart of Abjad's Symbolic Score-Control lies a powerful model that we call the Leaf Container Spanner Mark, or LCSM, model of the musical score.

The LCSM model can be schematically visualized as a superposition of two complementary and completely independent layers of structure: a *tree* that includes the Containers and the Leaves, and a layer of free floating *connectors* or *Spanners*.



There can be any number of Spanners, they may overlap, and they may connect to different levels of the tree hierarchy. The spanner attach to the elements of the tree, so a tree structure must exist for spanners to be made manifest.

3.1.1 Example 1

To understand the whys and hows of the LCSM model implemented in Abjad, it is probably easier to base the discussion on concrete musical examples. Let's begin with a simple and rather abstract musical fragment: a measure with nested tuplets.



What we see in this little fragment is a measure with 4/4 meter, 14 notes and four tuplet brackets prolating the notes. The three bottom tuplets (with ratios 5:4, 3:2, 5:4) prolates all but the last note. The topmost tuplet prolates all the notes in the measure and combines with the bottom three tuplets to doubly prolates all but the last note. The topmost tuplet as thus prolates three tuplets, each of which in turn prolates a group of notes. We can think of a tuplet as *containing* notes or other tuplets or both. Thus, in our example, the topmost tuplet contains three tuplets and a half note. Each of the tuplets contained by the topmost tuplet in turn contains five, three, and five notes respectively. If we add the measure, then we have a measure that contains a tuplet that contains tuplets that contain notes. The structure of the measure with nested tuplets as we have just described it has two important properties:

1. It is a *hierarchical* structure.
2. It follows *exclusive membership*, meaning that each element in the hierarchy (a note, a tuplet or a measure) has one and only one *parent*. In other words a single note is not contained in more than one tuplet simultaneously, and no one tuplet is contained in more than one other tuplet at the same time.

What we are describing here is a tree, and it is the structure of Abjad *containers*.

While this tree structure seem like the right way to represent the relationships between the elements of a score, it is not enough. Consider the tuplet example again with the following beaming alternatives:

Beaming alternative 1:



Beaming alternative 2:



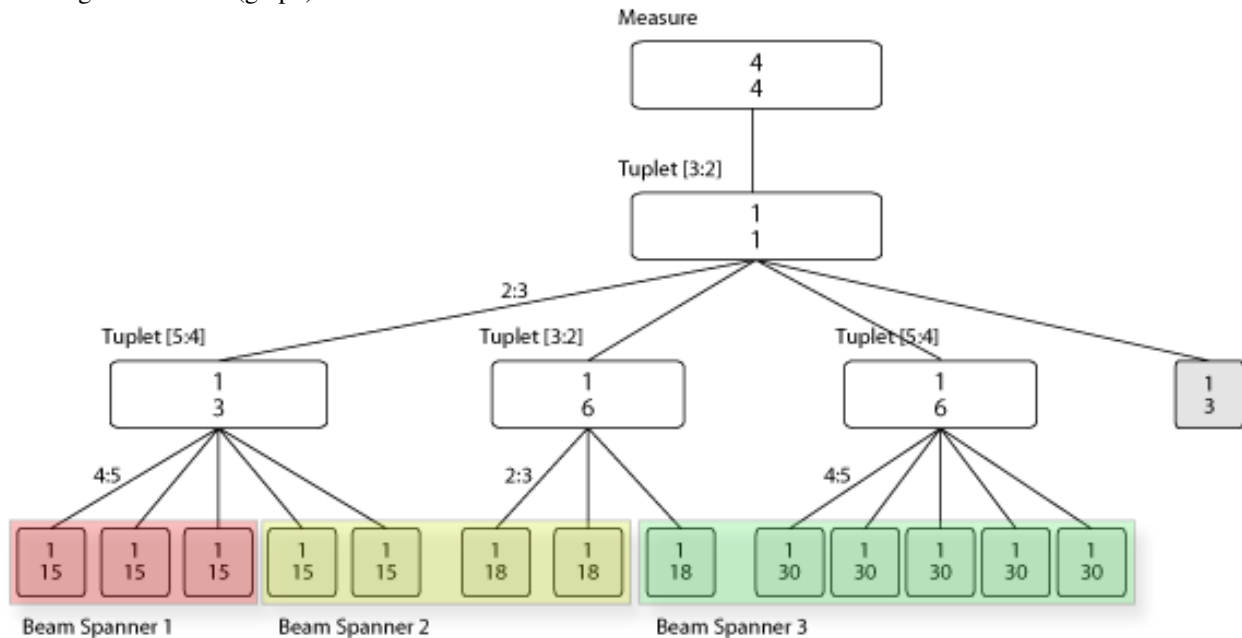
Beaming alternative 3:



Clearly the beaming of notes can be totally independent from the tuplet groupings. Beaming across tuplet groups implies beaming across nodes in the tree structure, which means that the beams do not adhere to the *exclusive (parent-hood) membership* characteristic of the tree. Beams must then be modeled independently as a separate and complementary structure. These are the Abjad *spanners*.

Below we have the score of our tuplet example with alternative beaming and its the Leaf-Container-Spanner graph. Notice that the colored blocks represent spanners.

Beaming alternative 3 (graph):



3.1.2 Example 2

As a second example let's look at the last five measures of Bartók's *Wandering* from *Mikrokosmos* vol. III. As simple as it may seem, these five measures carry with them a lot of information pertaining to musical notation.

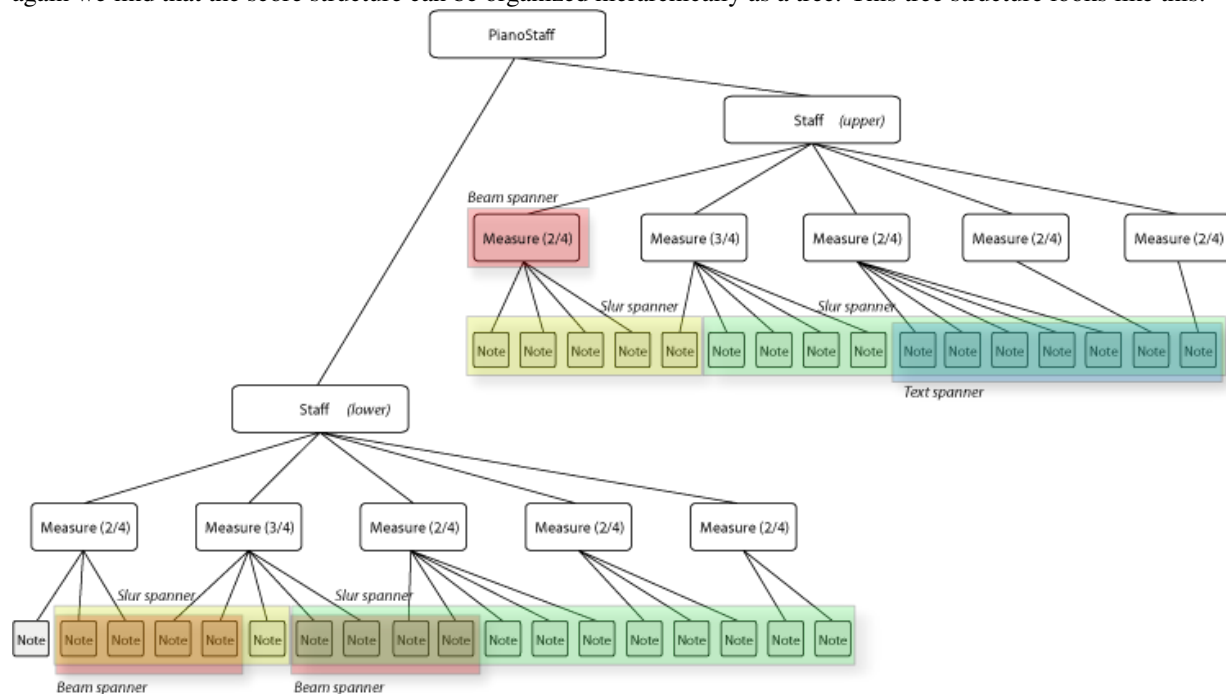


Note: Please refer to the [Bartok example](#) for a step by step construction of the musical fragment and its full Abjad code.

There are many musical signs of different types on the pages: notes, dynamic markings, clefs, staves, slurs, etc. These signs are structurally related to each other in different ways. Let's start by looking at the larger picture. The piano piece is written in two staves. As is customary, the staves are graphically grouped with a large curly brace attaching to them at the beginning or each system. Notice that each staff has a variety of signs associated with it. There are notes printed on the staff lines as well as meter indications and bar lines. Each note, for example, is in one and only one staff. A note is never in two staves at the same time. This is also true for measures. A measure in the top staff is not simultaneously drawn on the top staff and the bottom staff. It is better to think of each staff as having its own set of measures. Notice also that the notes in each staff fall within the region of one and only one measure, i.e. measures seem to contain notes. There is not one note that is at once in two measures (this is standard practice in musical notation, but it need not always be the case.)

As we continue describing the relationships between the musical signs in the page, we begin to discover a certain

structure, or a convenient way of structuring the score for conceptualization and manipulation. All the music in a piano score seems to be written in what we might call a *staff group*. The staff group is *composed of* two staves. Each staff in turn appears to be composed of a series of measures, and each measure is composed of a series of notes. So again we find that the score structure can be organized hierarchically as a tree. This tree structure looks like this:



Notice again though that there are elements in the score that imply and require a different kind of grouping. The two four eighth-note runs in the lower staff are beamed together across the bar line and, based on our tree structure, across tree nodes. So do the slurs, the dynamics markings and the ritardando indication at the top of the score. As we have seen in the tuplets example, all these groups running across the tree structure can be defined with *spanners*.

3.2 Parsing

Abjad provides a growing number of language parsers, including a sophisticated LilyPond parser,

3.2.1 LilyPond Parsing

`lilypondparsertools.LilyPondParser` parses a large, although incomplete, subset of LilyPond's syntax:

```
>>> from abjad.tools import lilypondparsertools
>>> parser = lilypondparsertools.LilyPondParser()
```

It understands notes, chords, skips and rests, including default durations and the `q` chord-repeat construct:

```
>>> string = r"{ c'\longa r4. <d' fs' bff'> g q8 s1 c'\breve. }"
>>> result = parser(string)

>>> f(result)
{
  c'\longa
  r4.
  <d' fs' bff'>4.
  g4.
}
```

```

<d' fs' bff'>8
s1
c''\breve.
}

```

```
>>> show(result)
```



It understands most spanners, articulations and dynamics too:

```

>>> string = r'''\new Staff {
...   c'8 \f \> (
...   d' -_ [
...   e' ^>
...   f' \ppp \<
...   g' \startTrillSpan \<
...   a' \)
...   b' ] \stopTrillSpan
...   c'' ) \accent \sfz
... }
... '''
>>> result = parser(string)

```

```

>>> f(result)
\new Staff {
  c'8 \f \> (
  d'8 -\portato [
  e'8 ^\accent
  f'8 \ppp \<
  g'8 \< \startTrillSpan
  a'8 \)
  b'8 ] \stopTrillSpan
  c''8 -\accent \sfz )
}

```

```
>>> show(result)
```



It understands contexts and markup:

```

>>> string = r'''\new Score <<
...   \new Staff = "Treble Staff" {
...     \new Voice = "Treble Voice" {
...       c' ^\markup { \bold Treble! }
...     }
...   }
...   \new Staff = "Bass Staff" {
...     \new Voice = "Bass Voice" {
...       \clef bass
...       c, _\markup { \italic Bass! }
...     }
...   }
... }

```

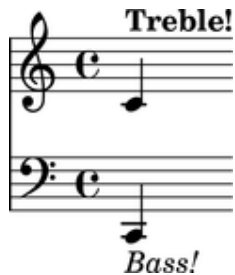
```

...     }
...     }
... >>
... '''
>>> result = parser(string)

>>> f(result)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/iotools/f.py", line 19, in f
    print expr.lilypond_format
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/contexttools/Context/Context.py", line 13, in
    return self._format_component()
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/componenttools/Component/Component.py", line 13, in
    result.extend(self._format_contents_slot(format_contributions))
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/containertools/Container/Container.py", line 13, in
    result.append(['contents', '_contents'], self._format_content_pieces())
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/containertools/Container/Container.py", line 13, in
    result.extend(m.lilypond_format.split('\n'))
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/contexttools/Context/Context.py", line 13, in
    return self._format_component()
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/componenttools/Component/Component.py", line 13, in
    result.extend(self._format_open_brackets_slot(format_contributions))
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/contexttools/Context/Context.py", line 13, in
    contributions = [context._format_invocation() + ' %s' % brackets_open[0]]
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/contexttools/Context/Context.py", line 13, in
    return r'\context %s = %s' % (self.context_name, schemetools.format_scheme_value(self.name))
NameError: global name 'schemetools' is not defined

```

```
>>> show(result)
```



It even understands certain aspects of LilyPond file layouts, like header blocks:

```

>>> string = r'''
... \header {
...   name = "Foo von Bar"
...   composer = \markup { by \bold \name }
...   title = \markup { The ballad of \name }
...   tagline = \markup { "" }
... }
... \score {
...   \new Staff {
...     \time 3/4
...     g' ( b' d'' )
...     e''4. ( c''8 c'4 )
...   }
... }
... '''
>>> result = parser(string)

```


by **Foo von Bar**

```
>>> string = r''\new Staff \relative c { c32 d e f g a b c d e f g a b c d e f g a b c }'''
>>> result = parser(string)
```

```
>>> f(result)
\new Staff {
    c32
    d32
    e32
    f32
    g32
    a32
    b32
    c'32
    d'32
    e'32
    f'32
    g'32
    a'32
    b'32
}
```

```
c'' 32
d'' 32
e'' 32
f'' 32
g'' 32
a'' 32
b'' 32
c''' 32
}

>>> show(result)
```



3.2.2 RhythmTree Parsing

`rhythmtreetools.RhythmTreeParser` parses a microlanguage resembling Ircam's RTM-style LISP syntax, and generates a sequence of `RhythmTree` structures, which can be further manipulated by composers, before being converted into Abjad score object:

```
>>> from abjad.tools import rhythmtreetools
>>> parser = rhythmtreetools.RhythmTreeParser()

>>> string = '(1 (1 (2 (1 1 1)) 2))'
>>> result = parser(string)
>>> result[0]
RhythmTreeContainer(
  children=(
    RhythmTreeLeaf(
      duration=1,
      is_pitched=True,
    ),
    RhythmTreeContainer(
      children=(
        RhythmTreeLeaf(
          duration=1,
          is_pitched=True,
        ),
        RhythmTreeLeaf(
          duration=1,
          is_pitched=True,
        ),
        RhythmTreeLeaf(
          duration=1,
          is_pitched=True,
        ),
      ),
      duration=2
    ),
    RhythmTreeLeaf(
      duration=2,
      is_pitched=True,
    ),
  ),
  duration=2
)
```

```

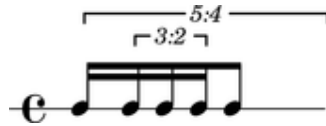
        ),
    ),
    duration=1
)

>>> tuplet = result[0]((1, 4))[0]
>>> f(tuplet)
\times 4/5 {
    c'16
    \times 2/3 {
        c'16
        c'16
        c'16
    }
    c'8
}

>>> staff = stafftools.RhythmicStaff([tuplet])

>>> show(staff, docs=True)

```



3.2.3 “Reduced-Ly” Parsing

`lilypondparsertools.ReducedLyParser` parses the “reduced-ly” microlanguage, whose syntax combines a very small subset of LilyPond syntax, along with affordances for generating various types of Abjad containers, and speedups for rapidly notating notes and rests without needing to specify pitches. It used mainly for creating Abjad documentation:

```

>>> from abjad.tools import rhythmtreetools
>>> parser = lilypondparsertools.ReducedLyParser()

>>> string = '| 4/4 c d e f || 3/8 r8 g4 |'
>>> result = parser(string)

>>> f(result)
{
  {
    \time 4/4
    c4
    d4
    e4
    f4
  }
  {
    \time 3/8
    r8
    g4
  }
}

```

```
>>> show(result)
```



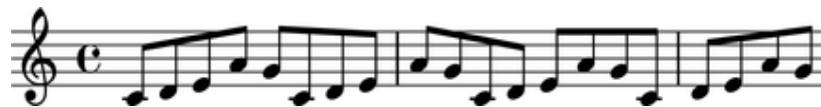
TUTORIALS

4.1 Changing notes to rests

4.1.1 A series of notes

It is easy to make a repeating pattern of notes.

```
>>> staff = Staff()
>>> staff.extend([Note(i, (1,8)) for i in [0, 2, 4, 6, 8] * 4])
>>> show(staff)
```



In the above example, a single list comprehension takes care of creating our notes.

4.1.2 Notes belonging to a staff can be iterated

We will create our repeated pattern again. Note that you can do this in one line:

```
>>> staff = Staff([Note(i, (1,8)) for i in [0, 2, 4, 6, 8] * 4])
```

Now, iterate over the staff's contents, substituting an eighth rest for every fifth (count from zero!) Note element in the staff:

```
>>> for i, note in enumerate(staff):
...     if (i%5) == 4:
...         staff[i] = Rest((1,8))
...
>>> show(staff)
```



4.1.3 Notes can be used directly.

In the previous example, we used an index (i) to keep track of where we are in the list of notes, and based our decision to flip on that index. We can also decide to flip a note to a rest based on the note itself:

```
>>> staff = Staff([Note(i, (1,8)) for i in [0, 2, 4, 9, 7] * 4 ])
>>> for i, note in enumerate(staff):
...     if note.sounding_pitch == "d'":
...         staff[i] = Rest((1,8))
...
>>> show(staff)
```



All the d's are now rests.

=tms

4.2 Creating rest-delimited slurs

Take a look at the slurs in the following example and notice that there is a pattern to how they arranged.



The pattern? Slurs in the example span groups of notes and chords separated by rests.

Abjad makes it easy to create such rest-delimited slurs in a structured way.

Let's start with the notes, rests and chords like this:

```
>>> string = r"\times 2/3 { c'4 d' r } r8 e'4 <fs' a' c''>8 ~ q4 \times 4/5 { r16 g' r b' d'' } df'4
>>> staff = Staff(string)
>>> show(staff)
```



Next we'll group notes and chords together with one of the functions available in the `componenttools` package, and add slur spanners inside our loop:

```
>>> leaves = iterationtools.iterate_leaves_in_expr(staff)
>>> for group in componenttools.yield_groups_of_mixed_klasses_in_sequence(leaves, (Note, Chord)):
...     spannertools.SlurSpanner(group)
...
SlurSpanner(c'4, d'4)
SlurSpanner(e'4, <fs' a' c''>8, <fs' a' c''>4)
SlurSpanner(g'16)
SlurSpanner(b'16, d''16, df'4, c'4, c'1)
```

And now we can take a look at the result:

```
>>> show(staff)
```



Notice now that there's a little problem with the notation we just created.

Four `SlurSpanners` were generated, but only three slurs are shown.

Why? LilyPond ignores slur terminators attached to the same note as the slur begins on.

Let's rewrite our example to prevent that from happening:

```
>>> staff = Staff(string)
>>> leaves = iterationtools.iterate_leaves_in_expr(staff)
>>> classes = (Note, Chord)
>>> for group in componenttools.yield_groups_of_mixed_klasses_in_sequence(leaves, classes):
...     if 1 < len(group):
...         spannertools.SlurSpanner(group)
...
SlurSpanner(c'4, d'4)
SlurSpanner(e'4, <fs' a' c''>8, <fs' a' c''>4)
SlurSpanner(b'16, d''16, df'4, c'4, c'1)
```

And now the corrected result:

```
>>> show(staff)
```



4.3 Making grob overrides

4.3.1 Grob-override component plug-ins

All Abjad containers have a grob-override plug-in:

```
>>> staff = Staff("c'4 d'4 e'4 f'4 g'4 a'4 g'2")
>>> staff.override.staff_symbol.color = 'blue'
>>> staff.override
LilyPondGrobOverrideComponentPlugIn(staff_symbol__color='blue')
```

All Abjad leaves have a grob-override plug-in, too:

```
>>> leaf = staff[-1]
>>> leaf.override.note_head.color = 'red'
>>> leaf.override.stem.color = 'red'
>>> leaf.override
LilyPondGrobOverrideComponentPlugIn(note_head__color='red', stem__color='red')
```

And so do Abjad spanners:

```
>>> slur = spannertools.SlurSpanner(staff[:])

>>> slur.override.slur.color = 'red'

>>> slur.override
LilyPondGrobOverrideComponentPlugIn(slur__color='red')
```

4.3.2 Grob proxies

Grob-override plug-ins contain grob proxies:

```
>>> leaf.override.note_head
LilyPondGrobProxy(color = 'red')

>>> leaf.override.stem
LilyPondGrobProxy(color = 'red')
```

4.3.3 Dot-chained override syntax

The's dot-chained grob override syntax shown here results from the special way that the Abjad grob-override plug-in and grob proxy set and get their attributes.

4.4 Understanding LilyPond grobs

LilyPond models music notation as a collection of graphic objects or grobs.

4.4.1 Grobs control typography

LilyPond grobs control the typographic details of the score:

```
>>> staff = Staff("c'4 ( d'4 ) e'4 ( f'4 ) g'4 ( a'4 ) g'2")

>>> f(staff)
\new Staff {
  c'4 (
  d'4 )
  e'4 (
  f'4 )
  g'4 (
  a'4 )
  g'2
}
```

```
>>> show(staff)
```



In the example above LilyPond creates a grob for every printed glyph. This includes the clef and time signature as well as the note heads, stems and slurs. If the example included beams, articulations or an explicit key signature then LilyPond would create grobs for those as well.

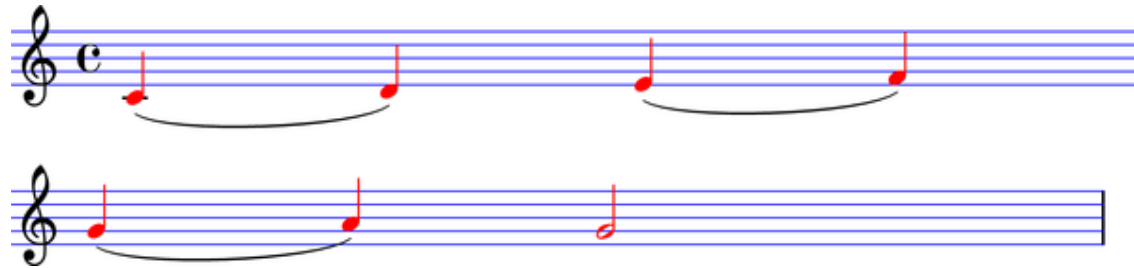
4.4.2 Grobs can be overridden

You can change the appearance of LilyPond grobs with grob overrides:

```
>>> staff.override.staff_symbol.color = 'blue'
>>> staff.override.note_head.color = 'red'
>>> staff.override.stem.color = 'red'

>>> f(staff)
\new Staff \with {
  \override NoteHead #'color = #red
  \override StaffSymbol #'color = #blue
  \override Stem #'color = #red
} {
  c'4 (
  d'4 )
  e'4 (
  f'4 )
  g'4 (
  a'4 )
  g'2
}

>>> show(staff, docs=True)
```



4.4.3 Nested Grob properties can be overridden

In the above example, *staff_symbol*, *note_head* and *stem* correspond to the LilyPond grobs *StaffSymbol*, *NoteHead* and *Stem*, while *color* in each case is the color properties of that graphic object.

It is not uncommon in LilyPond scores to see more complex overrides, consisting of a grob name and a list of two or more property names:

```
\override StaffGrouper #'staff-staff-spacing #'basic-distance = #7
```

To achieve the Abjad equivalent, simply concatenate the property names with double-underscores:

```
>>> staff = Staff()
>>> staff.override.staff_grouper.staff_staff_spacing__basic_distance = 7
>>> f(staff)
\new Staff \with {
  \override StaffGrouper #'staff-staff-spacing #'basic-distance = #7
} {
}
}
```

Abjad will explode the double-underscore delimited Python property into a LilyPond property list.

4.4.4 Check the LilyPond docs

New grobs are added to LilyPond from time to time.

For a complete list of LilyPond grobs see the [LilyPond documentation](#).

4.5 Understanding time signature marks

In this tutorial is to take a deeper look at what happens when we attach time signature marks to staves and other score components. To work through the tutorial, enter each of the examples into the Abjad interpreter and study what comes back. At the end of the tutorial you'll understand how time signature marks are created. You'll also understand how the states of different objects change when time signature marks are attached and detached.

First we start by creating a staff full of notes:

```
>>> staff = Staff("c'4 d'4 e'4 f'4 g'2")
```

If we ask the Abjad interpreter about our staff reference Abjad will respond with the interpreter display of the object:

```
>>> staff
Staff{5}
```

The 5 in `Staff{5}` shows that the staff contains 5 top-level components. The curly braces in `Staff{5}` show that the contents of the staff are to be read sequentially through time rather than in parallel.

Before we get to time signature marks let's take a moment and examine the state of the staff we've created. We can motivate this a bit by asking two questions:

1. what time signature is currently in effect for the staff we have just created?
2. **what is the time signature currently in effect for** the five notes contained within the staff we have just created?

The answer to both questions is the same: there is no time signature currently in effect for either our staff or for the five notes it contains.

We can see that this is the case with tools from the API:

```
>>> contexttools.get_effective_time_signature(staff) is None
True
```

And:

```
>>> for leaf in staff:
...     contexttools.get_effective_time_signature(leaf) is None
...
True
True
True
True
True
```

If we want, we can iterate both the staff and its leaves at one and the same time like this:

```
>>> for component in iterationtools.iterate_components_in_expr(staff):
...     component, contexttools.get_effective_time_signature(component)
...
(Staff{5}, None)
(Note("c'4"), None)
(Note("d'4"), None)
```

```
(Note("e'4"), None)
(Note("f'4"), None)
(Note("g'2"), None)
```

This confirms the answer to our questions that there is not yet any time signature in effect for any component in our staff because we have not yet attached a time signature mark to any component in our staff.

So what happens if we format our staff and send it off to LilyPond to render as a PDF? Will LilyPond render the staff with a time signature? Without a time signature? Will LilyPond refuse to render the example at all?

We find out like this:

```
>>> show(staff)
```



It turns out LilyPond defaults to a time signature of 4/4.

What's important to note here is that because we have not yet attached a time signature mark any component in our staff Abjad says “no effective time signature here” while LilyPond says “OK, I’ll default to 4/4 so we can get on with rendering your music.”

We can further confirm that this is the case by asking Abjad for the LilyPond format of our staff:

```
>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
  g'2
}
```

The LilyPond format of our staff contains no LilyPond `\time` command. This is, again, because we have not yet attached a time signature mark to any component in our staff.

We can now practice attaching and detaching time signature marks to different components in our staff and study what happens as we do.

We’ll start with 3/4.

The easiest thing to do is to attach a time signature mark to the staff itself.

We’ll do this in two separate steps and study each step to understand exactly what’s going on.

First, we create a 3/4 time signature mark:

```
>>> time_signature_mark = contexttools.TimeSignatureMark((3, 4))
```

If we ask the Abjad interpreter for the interpreter display of our time signature mark we get the following:

```
>>> time_signature_mark
TimeSignatureMark((3, 4))
```

All this tells us is that we have in fact created a 3/4 time signature mark. Nothing too exciting yet. At this point our 3/4 time signature is not yet attached to anything. We could say that the “state” of our time signature mark is “unattached.” And we can see this like so:

```
>>> time_signature_mark.start_component is None
True
```

What does it mean for a time signature mark to have 'start_component' equal to none? It means that the time signature isn't yet attached to any score component anywhere.

So now we attach our time signature mark to our staff:

```
>>> time_signature_mark.attach(staff)
TimeSignatureMark((3, 4))(Staff{5})
```

Abjad responds immediately by returning the time signature mark we have just attached.

Notice that our time signature mark's repr has changed. The repr of our 3/4 time signature mark now includes the repr of the staff to which we have just attached the time signature mark. That is to say that the repr of our time signature mark is `statal`.

Our time signature mark has transitioned from an "unattached" state to an "attached" state. We can see this like so:

```
>>> time_signature_mark.start_component
Staff{5}
```

And our staff has likewise transitioned from a state of having no effective time signature to a state of having an effective time signature:

```
>>> contexttools.get_effective_time_signature(staff)
TimeSignatureMark((3, 4))(Staff{5})
```

And what about the leaves inside our staff? Do the leaves now "know" that they are governed by a 3/4 time signature?

Indeed they do:

```
>>> for leaf in staff.leaves:
...     leaf, contexttools.get_effective_time_signature(leaf)
...
(Note("c'4"), TimeSignatureMark((3, 4))(Staff{5}))
(Note("d'4"), TimeSignatureMark((3, 4))(Staff{5}))
(Note("e'4"), TimeSignatureMark((3, 4))(Staff{5}))
(Note("f'4"), TimeSignatureMark((3, 4))(Staff{5}))
(Note("g'2"), TimeSignatureMark((3, 4))(Staff{5}))
```

So to briefly resume:

What we just did was to:

1. create a time signature mark
2. attach the time signature to a score component

This 2-step pattern is always the same when dealing with context marks: create then attach.

(We will find out later that there are short-cuts for different parts of this process. Right now we've chosen to create in a first step and attach in a second step so that we can examine the changing states of the objects involved.)

Before moving on let's look at the PDF corresponding to our staff:

```
>>> show(staff)
```



And let's confirm what we see in the PDF in the staff's format:

```
>>> f(staff)
\\new Staff {
    \\time 3/4
```

```

c' 4
d' 4
e' 4
f' 4
g' 2
}

```

The staff's format now contains a LilyPond `\time` command because we have attached an Abjad time signature mark to the staff.

What we've just been through above will cover over 80% of what you'll ever wind up doing with time signature marks: creating them and attaching them directly to staves. But what if we wanna get rid of a time signature mark? Or what if the time signature will be changing all over the place? We cover those cases next.

Detaching a time signature mark is easy:

```

>>> time_signature_mark.detach()
TimeSignatureMark((3, 4))

```

The Abjad returns the mark we have just detached. And, observing the repr of the time signature mark, we see that the time signature mark has again changed state: the time signature mark has transitioned from attached to unattached. We confirm this like so:

```

>>> time_signature_mark.start_component is None
True

```

And also like so:

```

>>> contexttools.get_effective_time_signature(staff) is None
True

```

Yup: our time signature mark knows nothing about our staff. And vice versa. This is good.

So now what if we want to set up a time signature of 2/4? That fits our music, too.

We have a couple of options.

We can simply create and attach a new time signature mark:

```

>>> duple_time_signature_mark = contexttools.TimeSignatureMark((2, 4))
>>> duple_time_signature_mark.attach(staff)
TimeSignatureMark((2, 4))(Staff{5})

>>> f(staff)
\new Staff {
  \time 2/4
  c' 4
  d' 4
  e' 4
  f' 4
  g' 2
}

>>> show(staff)

```



Yup. That works.

On the other hand, we could simply reuse our previous 3/4 time signature mark.

To do this we'll first detach our 2/4 time signature mark ...

```
>>> duple_time_signature_mark.detach()
TimeSignatureMark((2, 4))
```

... confirm that our staff is now time signatureless ...

```
>>> contexttools.get_effective_time_signature(staff) is None
True
```

```
>>> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

... reattach our previous 3/4 time signature ...

```
>>> time_signature_mark.attach(staff)
TimeSignatureMark((3, 4))(Staff{5})
```

... change the numerator of our time signature mark ...

```
>>> time_signature_mark.numerator = 2
```

... and check to make sure that everything is as it should be:

```
>>> contexttools.get_effective_time_signature(staff)
TimeSignatureMark((2, 4))(Staff{5})
>>> time_signature_mark.start_component
Staff{5}
```

```
>>> f(staff)
\new Staff {
    \time 2/4
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

```
>>> show(staff)
```



And everything works as it should.

To change to, for example, 4/4 we change just change the time signature mark's numerator again:

```
>>> time_signature_mark.numerator = 4
```

```
>>> f(staff)
\new Staff {
    \time 4/4
    c'4
}
```

```

    d' 4
    e' 4
    f' 4
    g' 2
}

```

But what if our time signature has a 2/4 pick-up?

The LilyPond command for pick-ups is `\partial`. Abjad time signature marks implement this as a read / write attribute:

```
>>> time_signature_mark.partial = Duration(2, 4)
```

```
>>> f(staff)
\new Staff {
  \partial 2
  \time 4/4
  c' 4
  d' 4
  e' 4
  f' 4
  g' 2
}

```

```
>>> show(staff)
```



And what if time signature changes all over the place?

We'll use the trivial example of a measure in 4/4 followed by a measure in 2/4.

To do this we will need two time signature marks.

We've already got a 4/4 time signature mark attached to our staff:

```
>>> f(staff)
\new Staff {
  \partial 2
  \time 4/4
  c' 4
  d' 4
  e' 4
  f' 4
  g' 2
}

```

Let's get rid of the pick-up:

```
>>> time_signature_mark.partial = None
```

```
>>> f(staff)
\new Staff {
  \time 4/4
  c' 4
  d' 4
  e' 4
  f' 4
}

```

```

    g' 2
}

```

Now what about the 2/4 time signature mark?

We create it in the usual way:

```

>>> duple_time_signature_mark = contexttools.TimeSignatureMark((2, 4))
>>> duple_time_signature_mark
TimeSignatureMark((2, 4))

```

But should we attach it? We can't attach our 2/4 time signature to our staff because we've already attached our 4/4 time signature to our staff. And it only makes sense to attach one time signature to any given score component.

Observe that we've built our score in a very straightforward way: we have a single staff that contains a (flat) sequence of notes. This means that we have only one choice for where to attach the new 2/4 time signature mark. And that is one the `g' 2` that comes on the downbeat of the second measure. We do that like this:

```

>>> duple_time_signature_mark.attach(staff[4])
TimeSignatureMark((2, 4))(g' 2)

```

```

>>> f(staff)
\new Staff {
  \time 4/4
  c' 4
  d' 4
  e' 4
  f' 4
  \time 2/4
  g' 2
}

```

```

>>> show(staff)

```



And everything works as we would like.

Incidentally, `staff[4]` means the component sitting at index 4 inside our staff. Using the interpreter we can verify that this is `g' 2`:

```

>>> staff[4]
Note("g' 2")

```

Depending on how we had chosen to build our staff we would have had more options for where to attach our 2/4 time signature mark. If, for example, we had chosen to populate our staff with a series of measures then it's possible we could have attached our 2/4 time signature to a measure instead of a note.

That covers the vast majority of things you'll do with time signature marks.

But before we stop we should mention another useful API function and then talk about some short-cuts.

First an API function to detach ALL context marks attaching to a component:

We call the function a first time:

```

>>> contexttools.detach_context_marks_attached_to_component(staff)
(TimeSignatureMark((4, 4)),)

```



```
>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
  \time 2/4
  g'2
}
```

And then a second time:

```
>>> contexttools.detach_context_marks_attached_to_component(staff[4])
(TimeSignatureMark((2, 4)),)

>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
  g'2
}
```

Now there are now context marks of any sort attached to our staff or to the notes in our staff.

Be careful with this function, though: it removes *all* context marks. So even though we just used the function to remove time signature marks, it also would have removed any clef marks or tempo marks if we had had those attached to our score, too.

And now for the short-cuts:

Our staff currently has no time signature marks attached:

```
>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
  g'2
}
```

So to recreate our 3/4 time signature we can do this ...

```
>>> time_signature_mark = contexttools.TimeSignatureMark((3, 4))
```

... and then use a short-cut to avoid calling `time_signature_mark.attach()` like this:

```
>>> time_signature_mark(staff)
TimeSignatureMark((3, 4))(Staff{5})

>>> f(staff)
\new Staff {
  \time 3/4
  c'4
  d'4
  e'4
  f'4
  g'2
}
```

What's going on here is that all context marks implement the special `__call__()` method as a short-cut for `attach()`. What is the special `__call__()` method? The `__call__()` method is what makes a function, class or any other Python object callable. The statement `time_signature_mark(staff)` has parentheses in it because the time signature mark is callable; and the time signature mark is callable because all context marks implement the special `__call__()` method.

Note too that all context marks understand an *empty call* as a short-cut for `detach()`. Like this:

```
>>> time_signature_mark()
TimeSignatureMark((3, 4))
```

```
>>> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

The empty call made against the time signature mark causes the time signature mark to detach from its start component.

The fact that context marks implement the special `__call__()` method as a short-cut for `attach()` means that context marks can be created and attached in a single line:

```
>>> contexttools.TimeSignatureMark((2, 4))(staff)
TimeSignatureMark((2, 4))(Staff{5})
```

```
>>> f(staff)
\new Staff {
    \time 2/4
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

What's going on here?

What's going on is that `contexttools.TimeSignatureMark((2, 4))` creates a time signature mark in the usual way and that – immediately after this – the newly created time signature mark is available for us to call it against our staff.

This last short-cut form of ...

```
>>> contexttools.TimeSignatureMark((2, 4))(staff)
```

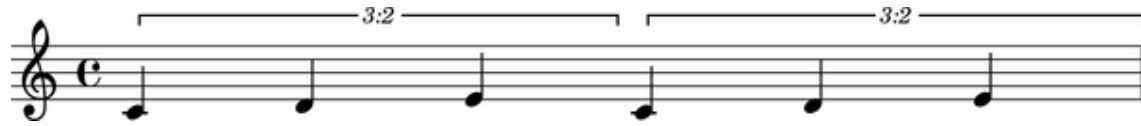
... is the usual way that you will see context marks of all sorts presented in the docs.

4.6 Working with component parentage

Many score objects contain other score objects.

```
>>> tuplet = Tuplet(Fraction(2, 3), "c'4 d'4 e'4")
>>> staff = Staff(2 * tuplet)
>>> score = Score([staff])
```

```
>>> show(score, docs=True)
```



Abjad uses the idea of parentage to model the way objects contain each other.

4.6.1 Improper parentage

The improper parentage of the first note in score begins with the note itself:

```
>>> note = score.leaves[0]

>>> componenttools.get_improper_parentage_of_component(note)
(Note("c'4"), Tuplet(2/3, [c'4, d'4, e'4]), Staff{2}, Score<<1>>)
```

4.6.2 Proper parentage

The proper parentage of the note begins with only the immediate parent of the note:

```
>>> componenttools.get_proper_parentage_of_component(note)
(Tuplet(2/3, [c'4, d'4, e'4]), Staff{2}, Score<<1>>)
```

Note: the length of the improper parentage of any component equals the length of the proper parentage of the component plus 1.

4.6.3 Parentage attributes

Use component tools to find score depth:

```
>>> componenttools.component_to_score_depth(note)
3
```

Or score root:

```
>>> componenttools.component_to_score_root(note)
Score<<1>>
```

Or to find whether a component has no (proper) parentage at all:

```
>>> componenttools.is_orphan_component(note)
False
```

4.7 Working with threads

4.7.1 What is a thread?

A thread is a structural relationship binding a set of strictly sequential voice-level components.

Threads may be explicitly defined via voice instances:

```
>>> v = Voice()
```

Or they may exist implicitly in certain score constructs in the absence of voice containers:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

Two contiguous voices must have the same name in order to be part of the same thread.

Here a thread does **not** exist between notes in different voices:

```
>>> v_one = Voice("c'16 d'16 e'16 f'16")
>>> v_two = Voice("c'8 d'8")
>>> staff = Staff([v_one, v_two])
>>> f(staff)
\new Staff {
  \new Voice {
    c'16
    d'16
    e'16
    f'16
  }
  \new Voice {
    c'8
    d'8
  }
}
```

Here a thread does exist:

```
>>> v_one.name = 'flute'
>>> v_two.name = 'flute'
>>> f(staff)
\new Staff {
  \context Voice = "flute" {
    c'16
    d'16
    e'16
    f'16
  }
  \context Voice = "flute" {
    c'8
    d'8
  }
}
```

4.7.2 What are threads for?

Consider the following situation:



Are the two eighth notes in the second half of the measure the continuation of the ascending line in the first half, or is it the quarter note? Is the very last C the continuation of the top melodic line or is it the A? The stems might suggest an answer, but for Abjad, stem direction is not structural. What path should Abjad take to traverse this little score from the first note to the last A? This same problem appears when trying to apply spanners to parallel structures. Thus,

threads are important in both score navigation and the application of spanners. In fact, threads are a requirement for spanner application.

In Abjad, the ambiguity is resolved through the explicit use of named voices.

The musical fragment above is constructed with the following code:

```
>>> vA = Voice(notetools.make_notes([5, 7, 9, 11], [(1, 8)] * 4))
>>> vB = Voice(notetools.make_notes([12, 11, 9], [(1, 8), (1, 8), (1, 4)]))
>>> vC = Voice(Note(12, (1, 4)) * 2)
>>> mark = marktools.LilyPondCommandMark('voiceOne')(vA[0])
>>> mark = marktools.LilyPondCommandMark('voiceOne')(vB[0])
>>> mark = marktools.LilyPondCommandMark('voiceTwo')(vC[0])
>>> p = Container([vB, vC])
>>> p.is_parallel = True
>>> staff = Staff([vA, p])

>>> f(staff)
\new Staff {
  \new Voice {
    \voiceOne
    f'8
    g'8
    a'8
    b'8
  }
  <<
    \new Voice {
      \voiceOne
      c''8
      b'8
      a'4
    }
    \new Voice {
      \voiceTwo
      c''4
      c''4
    }
  >>
}

>>> show(staff, docs=True)
```



There's a staff that sequentially contains a voice and a parallel container. The container in turn holds two voices running simultaneously.

It is now clear from the code that the last A belongs with the two descending eighth notes. But there's still no indication about a relationship of continuity between the first voice in the sequence (vA) and any of the two following voices. Note that, while the LilyPond voice number commands setting may suggest that vA and vB belong together, this is not the case. The LilyPond voice number commands simply set the direction of stems in printed output.

To see this more clearly, suppose we want to add a slur spanner starting on the first note and ending on one of the last simultaneous notes. To attach the slur spanner to the voices we could try either:

```
>>> spannertools.SlurSpanner([vA, vB])
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/spannertools/SlurSpanner/SlurSpanner.py", line 10, in <module>
    DirectedSpanner.__init__(self, components, direction)
File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/spannertools/DirectedSpanner/DirectedSpanner.py", line 10, in <module>
```



```
    Spanner.__init__(self, components)
File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/spannertools/Spanner/Spanner.py", line 1
```



```

    self._initialize_components(components)
File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/spannertools/Spanner/Spanner.py", line 11, in
    assert componenttools.all_are_thread_contiguous_components(leaves)
AssertionError

... Or ...

>>> spannertools.SlurSpanner([vA, vC])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/spannertools/SlurSpanner/SlurSpanner.py", line 11, in
    DirectedSpanner.__init__(self, components, direction)
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/spannertools/DirectedSpanner/DirectedSpanner.py", line 11, in
    Spanner.__init__(self, components)
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/spannertools/Spanner/Spanner.py", line 11, in
    self._initialize_components(components)
  File "/Users/josiah/Documents/Projects/abjad/trunk/abjad/tools/spannertools/Spanner/Spanner.py", line 11, in
    assert componenttools.all_are_thread_contiguous_components(leaves)
AssertionError

```

But both raise a contiguity error. Abjad needs to see an explicit connection between either vA and vB or between vA and vC .

Observe the behavior of the `iterate_thread_in_expr()` iterator on the *staff*:

```

>>> vA_thread_signature = componenttools.component_to_containment_signature(vA)
>>> notes = iterationtools.iterate_thread_in_expr(staff, Note, vA_thread_signature)
>>> print list(notes)
[Note("f'8"), Note("g'8"), Note("a'8"), Note("b'8")]

>>> vB_thread_signature = componenttools.component_to_containment_signature(vB)
>>> notes = iterationtools.iterate_thread_in_expr(staff, Note, vB_thread_signature)
>>> print list(notes)
[Note("c'8"), Note("b'8"), Note("a'4")]

>>> vC_thread_signature = componenttools.component_to_containment_signature(vC)
>>> notes = iterationtools.iterate_thread_in_expr(staff, Note, vC_thread_signature)
>>> print list(notes)
[Note("c'4"), Note("c'4")]

```

In each case we are passing a different **thread signature** to the `iterate_thread_in_expr()` iterator, so each case returns a different list of notes.

We can see that the thread signature of each voice is indeed different by printing it:

```

>>> vA_thread_signature = componenttools.component_to_containment_signature(vA)
>>> vA_thread_signature
ContainmentSignature(Voice-25493808, Voice-25493808, Staff-25494192)

>>> vB_thread_signature = componenttools.component_to_containment_signature(vB)
>>> vB_thread_signature
ContainmentSignature(Voice-25493936, Voice-25493936, Staff-25494192)

>>> vC_thread_signature = componenttools.component_to_containment_signature(vC)
>>> vC_thread_signature
ContainmentSignature(Voice-25494064, Voice-25494064, Staff-25494192)

```

And by comparing them with the binary equality operator:

```
>>> vA_thread_signature == vB_thread_signature
False
>>> vA_thread_signature == vC_thread_signature
False
>>> vB_thread_signature == vC_thread_signature
False
```

To allow Abjad to treat the content of, say, voices *vA* and *vB* as belonging together, we explicitly define a thread between them. To do this all we need to do is give both voices the same name:

```
>>> vA.name = 'piccolo'
>>> vB.name = 'piccolo'
```

Now *vA* and *vB* and all their content belong to the same thread:

```
>>> vA_thread_signature == vB_thread_signature
False
```

Note how the thread signatures have changed:

```
>>> vA_thread_signature = componenttools.component_to_containment_signature(vA)
>>> print vA_thread_signature
staff: Staff-25494192
voice: Voice-'piccolo'
self: Voice-'piccolo'

>>> vB_thread_signature = componenttools.component_to_containment_signature(vB)
>>> print vB_thread_signature
staff: Staff-25494192
voice: Voice-'piccolo'
self: Voice-'piccolo'

>>> vC_thread_signature = componenttools.component_to_containment_signature(vC)
>>> print vC_thread_signature
staff: Staff-25494192
voice: Voice-25494064
self: Voice-25494064
```

And how the `iterationtools.iterate_thread_in_expr()` function returns all the notes belonging to both *vA* and *vB* when passing it the full staff and the thread signature of *vA*:

```
>>> notes = iterationtools.iterate_thread_in_expr(staff, Note, vA_thread_signature)
>>> print list(notes)
[Note("f'8"), Note("g'8"), Note("a'8"), Note("b'8"), Note("c''8"), Note("b'8"), Note("a'4")]
```

Now the slur spanner can be applied to voices *vA* and *vB*:

```
>>> spannertools.SlurSpanner([vA, vB])
SlurSpanner({f'8, g'8, a'8, b'8}, {c''8, b'8, a'4})
```

or directly to the notes returned by the `iterate_thread_in_expr()` iteration tool, which are the notes belonging to both *vA* and *vB*:

```
>>> notes = iterationtools.iterate_thread_in_expr(staff, Note, vA_thread_signature)
>>> spannertools.SlurSpanner(list(notes))
SlurSpanner(f'8, g'8, a'8, b'8, c''8, b'8, a'4)

>>> show(staff, docs=True)
```



4.7.3 Coda

We could have constructed this score in a simpler way with only two voices, one of them starting with a LilyPond skip:

```
>>> vX = Voice(notetools.make_notes([5, 7, 9, 11, 12, 11, 9], [(1, 8)] * 6 + [(1, 4)]))
>>> vY = Voice([skiptools.Skip((2, 4))] + Note(12, (1, 4)) * 2)
>>> mark = marktools.LilyPondCommandMark('voiceOne')(vX[0])
>>> mark = marktools.LilyPondCommandMark('voiceTwo')(vY[0])
>>> staff = Staff([vX, vY])
>>> staff.is_parallel = True

>>> f(staff)
\new Staff <<
  \new Voice {
    \voiceOne
    f'8
    g'8
    a'8
    b'8
    c''8
    b'8
    a'4
  }
  \new Voice {
    \voiceTwo
    s2
    c''4
    c''4
  }
>>

>>> show(staff, docs=True)
```



REFERENCE MANUAL

5.1 Annotations

Annotate components with user-specific information for future use.

Annotations do not impact formatting.

5.1.1 Creating annotations

Use mark tools to create annotations:

```
>>> annotation = marktools.Annotation('special pitch', pitchtools.NamedChromaticPitch('bs'))

>>> annotation
Annotation('special pitch', NamedChromaticPitch('bs'))
```

5.1.2 Attaching annotations to a component

Attach annotations to any component with `attach()`:

```
>>> note = Note("c'4")
>>> annotation.attach(note)
Annotation('special pitch', NamedChromaticPitch('bs')) (c'4)

>>> annotation
Annotation('special pitch', NamedChromaticPitch('bs')) (c'4)

>>> another_annotation = marktools.Annotation('special pitch', pitchtools.NamedChromaticPitch('bs'))
>>> another_annotation.attach(note)
Annotation('special pitch', NamedChromaticPitch('bs')) (c'4)

>>> another_annotation
Annotation('special pitch', NamedChromaticPitch('bs')) (c'4)
```

5.1.3 Getting the annotations attached to a component

Use mark tools to get all the annotations attached to a component:

```
>>> marktools.get_annotations_attached_to_component(note)
(Annotation('special pitch', NamedChromaticPitch('bs'))(c'4), Annotation('special pitch', NamedChromaticPitch('bs'))(c'4))
```

5.1.4 Detaching annotations from a component one at a time

Use `detach()` to detach annotations from a component one at a time:

```
>>> annotation.detach()
Annotation('special pitch', NamedChromaticPitch('bs'))

>>> annotation
Annotation('special pitch', NamedChromaticPitch('bs'))
```

5.1.5 Detaching all annotations attached to a component at once

Or use `marktools` to detach all annotations attached to a component at once:

```
>>> print marktools.detach_annotations_attached_to_component(note)
(Annotation('special pitch', NamedChromaticPitch('bs')),)

>>> marktools.get_annotations_attached_to_component(note)
()
```

5.1.6 Inspecting the component to which an annotation is attached

Use `start_component` to inspect the component to which an annotation is attached:

```
>>> annotation.attach(note)
Annotation('special pitch', NamedChromaticPitch('bs'))(c'4)

>>> annotation.start_component
Note("c'4")
```

5.1.7 Inspecting annotation name

Use `name` to get the name of any annotation:

```
>>> annotation.name
'special pitch'
```

5.1.8 Inspecting annotation value

And use `value` to get the value of any annotation:

```
>>> annotation.value
NamedChromaticPitch('bs')
```

5.2 Articulations

Articulations model staccati, marcati, tenuti and other symbols. Articulations attach notes, rests or chords.

5.2.1 Creating articulations

Use `marktools` to create articulations:

```
articulation = marktools.Articulation('turn')
```

```
>>> articulation
Articulation('turn')
```

5.2.2 Attaching articulations to a leaf

Use `attach()` to attach articulations to a leaf:

```
>>> staff = Staff([])
>>> key_signature = contexttools.KeySignatureMark('g', 'major')
>>> key_signature.attach(staff)
time_signature = contexttools.TimeSignatureMark((2, 4), partial = Duration(1, 8))
>>> time_signature.attach(staff)

>>> staff.extend("d'8 f'8 a'8 d''8 f''8 gs'4 r8 e'8 gs'8 b'8 e''8 gs''8 a'4")

>>> articulation.attach(staff[5])

>>> show(staff)
```



(The example is based on Haydn's piano sonata number 42, Hob. XVI/27.)

5.2.3 Attaching articulations to many notes and chords at once

Use `marktools` to attach articulations to many notes and chords at one time:

```
>>> marktools.attach_articulations_to_notes_and_chords_in_expr(staff[:6], ['.''])
>>> show(staff)
```



5.2.4 Getting the articulations attached to a leaf

Use `marktools` to get the articulations attached to a leaf:

```
>>> marktools.get_articulations_attached_to_component(staff[5])
(Articulation('turn')(gs'4), Articulation('.') (gs'4))
```

5.2.5 Detaching articulations from a leaf one at a time

Detach articulations by hand with `detach()`:

```
>>> articulation.detach()
```

```
>>> articulation
-\turn
```

```
>>> show(staff)
```



5.2.6 Detaching all articulations attached to a leaf at once

Use `marktools` to detach all articulations attached to a leaf at once:

```
>>> staff[0]
Note("d'8")
```

```
>>> marktools.detach_articulations_attached_to_component(staff[0])
```

```
>>> show(staff)
```



5.2.7 Inspecting the leaf to which an articulation is attached

Use `start_component` to inspect the component to which an articulation is attached:

```
>>> articulation = marktools.Articulation('turn')
>>> articulation.attach(staff[-1])
```

```
>>> show(staff)
```



```
>>> articulation.start_component
Note("a'4")
```

5.2.8 Understanding the interpreter display of an articulation that is not attached to a leaf

The interpreter display of an articulation that is not attached to a leaf contains three parts:

```
>>> articulation = marktools.Articulation('staccato')
```

```
>>> articulation
>>> print repr(articulation)
Articulation('staccato')
```

Articulation tells you the articulation's class.

'staccato' tells you the articulation's name.

If you set the direction string of the articulation then that will appear, too:

```
>>> articulation.direction_string = '^'

>>> articulation
>>> print repr(articulation)
Articulation('staccato', '^')
```

5.2.9 Understanding the interpreter display of an articulation that is attached to a leaf

The interpreter display of an articulation that is attached to a leaf contains four parts:

```
>>> articulation.attach(staff[-1])

>>> articulation
>>> print repr(articulation)
Articulation('staccato', '^') (a'4)

>>> show(staff)
```



Articulation tells you the articulation's class.

'staccato' tells you the articulation's name.

'^' tells you the articulation's direction string.

(a"4) tells you the component to which the articulation is attached.

If you set the direction string of the articulation to none then the direction will no longer appear:

```
>>> articulation.direction_string = None

>>> articulation
Articulation('staccato') (a'4)
```

5.2.10 Understanding the string representation of an articulation

The string representation of an articulation comprises two parts:

```
>>> str(articulation)
'-\\staccato'
```

- tells you the articulation's direction string.

staccato tells you the articulation's name.

5.2.11 Inspecting the LilyPond format of an articulation

Get the LilyPond input format of an articulation with `format`:

```
>>> articulation.format
'-\\staccato'
```

Use `f()` as a short-cut to print the LilyPond format of an articulation:

```
>>> f(articulation)
-\\staccato
```

5.2.12 Controlling whether an articulation appears above or below the staff

Set `direction_string` to `'^'` to force an articulation to appear above the staff:

```
>>> articulation.direction_string = '^'

>>> show(staff)
```



Set `direction_string` to `'_'` to force an articulation to appear below the staff:

```
>>> articulation.direction_string = '_'

>>> show(staff)
```



Set `direction_string` to `None` to allow LilyPond to position an articulation automatically:

```
>>> articulation.direction_string = None

>>> show(staff)
```



5.2.13 Getting and setting the name of an articulation

Set the name of an articulation to change the symbol an articulation prints:

```
>>> articulation.name = 'staccatissimo'

>>> show(staff)
```



5.2.14 Copying articulations

Use `copy.copy()` to copy an articulation:

```
>>> import copy

>>> articulation_copy_1 = copy.copy(articulation)

>>> articulation_copy_1
Articulation('staccatissimo')

>>> articulation_copy_1.attach(staff[1])

>>> show(staff)
```



Or use `copy.deepcopy()` to do the same thing.

5.2.15 Comparing articulations

Articulations compare equal with equal direction names and direction strings:

```
>>> articulation.name
'staccatissimo'
>>> articulation.direction_string
None

>>> articulation_copy_1.name
'staccatissimo'
>>> articulation_copy_1.direction_string
None

>>> articulation == articulation_copy_1
True
```

Otherwise articulations do not compare equal.

5.2.16 Overriding attributes of the LilyPond script grob

Override attributes of the LilyPond script grob like this:

```
>>> staff.override.script.color = 'red'

>>> f(staff)
\new Staff \with {
  \override Script #'color = #red
} {
  \key g \major
  \partial 8
  \time 2/4
  d'8
  f'8 -\staccatissimo -\staccato
  a'8 -\staccato
```

```

d''8 -\staccato
f''8 -\staccato
gs'4 -\staccato
r8
e'8
gs'8
b'8
e''8
gs''8
a'4 -\staccatissimo -\turn
}

>>> show(staff)

```



See the LilyPond documentation for a list of script grob attributes available.

5.3 Chords

5.3.1 Making chords from a LilyPond input string

You can make chords from a LilyPond input string:

```

>>> chord = Chord("<c' d' bf'>4")
>>> show(chord, docs=True)

```



5.3.2 Making chords from chromatic pitch numbers and duration

You can also make chords from chromatic pitch numbers and duration:

```

>>> chord = Chord([0, 2, 10], Duration(1, 4))
>>> show(chord, docs=True)

```



5.3.3 Getting all the written pitches of a chord at once

You can get all the written pitches of a chord at one time:

```

>>> chord.written_pitches
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"), NamedChromaticPitch("bf'"))

```

Abjad returns a read-only tuple of named chromatic pitches.

5.3.4 Getting the written pitches of a chord one at a time

You can get the written pitches of a chord one at a time:

```
>>> chord.written_pitches[0]
NamedChromaticPitch("c' ")
```

Chords index the pitch they contain starting from 0 (just like tuples and lists).

5.3.5 Adding one pitch to a chord at a time

Use `append()` to add one note to a chord.

You can add a pitch to a chord with a chromatic pitch number:

```
>>> chord.append(9)

>>> show(chord, docs=True)
```



Or you can add a pitch to a chord with a chromatic pitch name:

```
>>> chord.append("df' ")

>>> show(chord, docs=True)
```



Chords sort their pitches every time you add a new one.

This means you can add pitches to your chord in any order.

5.3.6 Adding many pitches to a chord at once

Use `extend()` to add many pitches to a chord.

You can use chromatic pitch numbers:

```
>>> chord.extend([3, 4, 14])

>>> show(chord, docs=True)
```



Or you can use chromatic pitch names:

```
>>> chord.extend(["g' ", "af' "])

>>> show(chord, docs=True)
```



5.3.7 Deleting pitches from a chord

Delete pitches from a chord with `del()`:

```
>>> del(chord[0])
```

```
>>> show(chord, docs=True)
```



```
>>> del(chord[0])
```

```
>>> show(chord, docs=True)
```



Negative indices work too:

```
>>> del(chord[-1])
```

```
>>> show(chord, docs=True)
```



5.3.8 Formatting chords

Get the LilyPond input format of any Abjad object with `format`:

```
>>> chord.format
<ef' e' a' bf' df'' d'' g''>4
```

Use `f()` as a short-cut to print the LilyPond input format of any Abjad object:

```
>>> f(chord)
<ef' e' a' bf' df'' d'' g''>4
```

5.3.9 Working with note heads

Most of the time you will work with the pitches of a chord. But you can get the note heads of a chord, too:


```
>>> chord.note_heads
(NoteHead("ef'"), NoteHead("e'"), NoteHead("a'"), NoteHead("bf'"), NoteHead("df''"), NoteHead("d''"))
```

This is useful when you want to apply LilyPond overrides to note heads in a chord one at a time:

```
>>> chord[2].tweak.color = 'red'
>>> chord[3].tweak.color = 'blue'
>>> chord[4].tweak.color = 'green'
```

```
>>> f(chord)
<
    ef'
    e'
    \tweak #'color #red
    a'
    \tweak #'color #blue
    bf'
    \tweak #'color #green
    df''
    d''
    g''
>4
```

```
>>> show(chord, docs=True)
```



5.3.10 Working with empty chords

Abjad allows empty chords:

```
>>> chord = Chord([], Duration(1, 4))
Chord('<>4')
```

Abjad formats empty chords, too:

```
>>> f(chord)
<>4
```

But if you pass empty chords to `show()` LilyPond will complain because empty chords don't constitute valid LilyPond input.

When you are done working with an empty chord you can add pitches back into it chord in any of the ways described above:

```
>>> chord.extend(["gf'", "df''", "g''"])
```

```
>>> show(chord, docs=True)
```



5.4 Containers

5.4.1 Creating containers

Create a container with components:

```
>>> container = Container([Note("ds'16"), Note("cs'16"), Note("e'16"), Note("c'16")])
>>> show(container)
```



Or with a note-entry string:

```
>>> container = Container("ds'16 cs'16 e'16 c'16 d'2 ~ d'8")
>>> show(container)
```



5.4.2 Inspecting music

Return the components in a container with `music`:

```
>>> container.music
(Note("ds'16"), Note("cs'16"), Note("e'16"), Note("c'16"), Note("d'2"), Note("d'8"))
```

Or with a special call to `__getslice__`:

```
>>> container[:]
[Note("ds'16"), Note("cs'16"), Note("e'16"), Note("c'16"), Note("d'2"), Note("d'8")]
```

5.4.3 Inspecting length

Get the length of a container with `len()`:

```
>>> len(container)
6
```

5.4.4 Inspecting duration

Contents duration equals the sum of the duration of everything inside the container:

```
>>> container.contents_duration
Duration(7, 8)
```

5.4.5 Adding one component to the end of a container

Add one component to the end of a container with `append`:

```
>>> container.append(Note("af' 32"))
```

```
>>> show(container)
```



5.4.6 Adding many components to the end of a container

Add many components to the end of a container with `extend`:

```
>>> container.extend([Note("c' 32"), Note("a' 32")])
```

```
>>> show(container)
```



5.4.7 Finding the index of a component

Find the index of a component with `index`:

```
>>> note = container[7]
```

```
>>> container.index(note)
```

7

5.4.8 Inserting a component by index

Insert a component by index with `insert`:

```
>>> container.insert(-3, Note("g'32"))
```

```
>>> show(container)
```



5.4.9 Removing a component by index

Remove a component by index with `pop`:

```
>>> container.pop(-1)
```

```
>>> show(container)
```



5.4.10 Removing a component by reference

Remove a component by reference with `remove`:

```
>>> container.remove(container[-1])
```

```
>>> show(container)
```



Note: `__getslice__`, `__setslice__` and `__delslice__` remain to be documented.

5.4.11 Naming containers

You can name Abjad containers:

```
>>> flute_staff = Staff("c'8 d'8 e'8 f'8")
>>> flute_staff.name = 'Flute'
>>> violin_staff = Staff("c'8 d'8 e'8 f'8")
>>> violin_staff.name = 'Violin'
>>> staff_group = scoretools.StaffGroup([flute_staff, violin_staff])
>>> score = Score([staff_group])
```

Container names appear in LilyPond input:

```
>>> f(score)
\new Score <<
  \new StaffGroup <<
    \context Staff = "Flute" {
      c'8
      d'8
      e'8
      f'8
    }
    \context Staff = "Violin" {
      c'8
      d'8
      e'8
      f'8
    }
  >>
>>
```

And make it easy to retrieve containers later:

```
>>> componenttools.get_first_component_in_expr_with_name(score, 'Flute')
Staff-"Flute"{4}
```

But container names do not appear in notational output:

```
>>> show(score)
```



5.4.12 Understanding { } and << >> in LilyPond

LilyPond uses curly { } braces to wrap a stream of musical events that are to be engraved one after the other:

```
\new Voice {
  e''4
  f''4
  g''4
  g''4
  f''4
  e''4
  d''4
  d''4 \fermata
}
```



LilyPond uses skeleton << >> braces to wrap two or more musical expressions that are to be played at the same time:

```
\new Staff <<
  \new Voice {
    \voiceOne
    e''4
    f''4
    g''4
    g''4
    f''4
    e''4
    d''4
    d''4 \fermata
  }
  \new Voice {
    \voiceTwo
    c''4
    c''4
    b'4
    c''4
    c''8
    b'8
    c''4
  }
>>
```

```

        b'4
        b'4 \fermata
    }
>>

```



The examples above are both LilyPond input.

The most common use of LilyPond { } is to group a potentially long stream of notes and rests into a single expression.

The most common use of LilyPond << >> is to group a relatively smaller number of note lists together polyphonically.

5.4.13 Understanding sequential and parallel containers

Abjad implements LilyPond { } and << >> in the container `is_parallel` attribute.

Some containers set `is_parallel` to false at initialization:

```

staff = Staff([])
staff.is_parallel
False

```

Other containers set `is_parallel` to true:

```

score = Score([])
score.is_parallel
True

```

5.4.14 Changing sequential and parallel containers

Set `is_parallel` by hand as necessary:

```

voice_1 = Voice(r"e''4 f''4 g''4 g''4 f''4 e''4 d''4 d''4  ermata")
voice_2 = Voice(r"c''4 c''4 b'4 c''4 c''8 b'8 c''4 b'4 b'4  ermata")
>>> staff = Staff([voice_1, voice_2])
>>> staff.is_parallel = True
>>> marktools.LilyPondCommandMark('voiceOne')(voice_1)
>>> marktools.LilyPondCommandMark('voiceTwo')(voice_2)
>>> show(staff)

```



The staff in the example above is set to parallel after initialization to create a type of polyphonic staff:

```

>>> f(staff)
\new Staff <<
    \new Voice {
        \voiceOne
        e''4
        f''4
    }
>>>

```

```

        g''4
        g''4
        f''4
        e''4
        d''4
        d''4 -\fermata
    }
    \new Voice {
        \voiceTwo
        c''4
        c''4
        b'4
        c''4
        c''8
        b'8
        c''4
        b'4
        b'4 -\fermata
    }
>>

```

5.4.15 Overriding containers

The symbols below are black with fixed thickness and predetermined spacing:

```

>>> staff = Staff("c'4 d'4 e'4 f'4 g'4 a'4 g'2")
>>> slur_1 = spannertools.SlurSpanner(staff[:2])
>>> slur_2 = spannertools.SlurSpanner(staff[2:4])
>>> slur_3 = spannertools.SlurSpanner(staff[4:6])

>>> f(staff)
\new Staff {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    g'2
}

>>> show(staff)

```



But you can override LilyPond grobs to change the look of Abjad containers:

```

>>> staff.override.staff_symbol.color = 'blue'

>>> f(staff)
\new Staff \with {
    \override StaffSymbol #'color = #blue
} {
    c'4 (
    d'4 )
}

```

```
e'4 (  
f'4 )  
g'4 (  
a'4 )  
g'2  
}  
  
>>> show(staff)
```



5.4.16 Overriding containers' contents

You can override LilyPond grobs to change the look of containers' contents, too:

```
>>> staff.override.note_head.color = 'red'  
>>> staff.override.stem.color = 'red'  
  
>>> f(staff)  
\new Staff \with {  
  \override NoteHead #'color = #red  
  \override StaffSymbol #'color = #blue  
  \override Stem #'color = #red  
} {  
  c'4 (  
  d'4 )  
  e'4 (  
  f'4 )  
  g'4 (  
  a'4 )  
  g'2  
}  
  
>>> show(staff)
```



5.4.17 Removing container overrides

Delete grob overrides you no longer want:

```
>>> del(staff.override.staff_symbol)  
  
>>> f(staff)  
\new Staff \with {  
  \override NoteHead #'color = #red  
  \override Stem #'color = #red  
} {  
  c'4 (  
  d'4 )  
  e'4 (  
  f'4 )  
  g'4 (  
  a'4 )  
  g'2  
}
```



```

    f' 4 )
    g' 4 (
    a' 4 )
    g' 2
}

>>> show(staff)

```



5.5 Durations

5.5.1 Introduction

Abjad publishes information about many durated score objects.

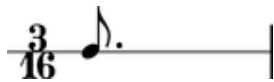
Notes, rests, chords and skips carry some duration attributes:

```

>>> note = Note(0, (3, 16))
>>> measure = Measure((3, 16), [note])
>>> staff = stafftools.RhythmicStaff([measure])

>>> show(staff, docs=True)

```



```

>>> note.written_duration
Duration(3, 16)

```

Tuplets, measures, voices, staves and the other containers carry duration attributes, too:

```

>>> tuplet = tuplettools.FixedDurationTuplet(Duration(3, 16), "c'16 c' c' c' c'")
>>> measure = Measure((3, 16), [tuplet])
>>> staff = stafftools.RhythmicStaff([measure])

>>> show(staff, docs=True)

```



```

>>> tuplet.multiplier
Fraction(3, 5)

```

The next chapters document core duration concepts in Abjad.

5.5.2 Assignability

Western notation readily admits rational values like $1/4$. But values like $1/5$ notate only with triplet brackets or special time signatures. Abjad formalizes the difference between rationals like $1/4$ and $1/5$ in the definition of rational assignability.

Rational values n/d are assignable when and only when numerator n is of the form $k(2^{**u}-j)$ and denominator d is of the form 2^{**v} . In this definition u and v must be nonnegative integers, k must be a positive integer, and j must be either 0 or 1.

Abjad initializes notes, rests and chords with assignable durations only.

5.5.3 Prolation

Abjad uses prolation as a cover term for rhythmic augmentation and diminution. Augmentation increases the duration of notes, rests and chords. Diminution does the opposite. Western notation employs tuplet brackets and special types of time signature to effect prolation.

Tuplet prolation

Tuplets prolate their contents:

```
>>> tuplet = Tuplet(Fraction(5, 4), 'c8 c c c')
>>> staff = stafftools.RhythmicStaff([Measure((5, 8), [tuplet])])
>>> beam = beamtools.BeamSpanner(tuplet)
```

```
>>> show(staff, docs=True)
```



```
>>> note = tuplet[0]
>>> note.written_duration
Duration(1, 8)
```

```
>>> note.prolation
Fraction(5, 4)
```

```
>>> note.prolated_duration
Duration(5, 32)
```

Notes here with written duration $1/8$ carry prolation factor $5/4$ and prolated duration $5/32$.

Western notation does not recognize tuplet brackets carrying one-to-one ratios. Such trivial tuplets may, however, be useful during different stages of composition, and Abjad allows them for that reason. Trivial tuplets carry zero prolation. Zero-prolated tuplets neither augment nor diminish the music they contain.

Meter prolation

Time signatures in western notation usually carry a denominator equal to a nonnegative integer power of 2. Abjad calls these conventional meters binary meters. Denominators equal to integers other than integer powers of 2 are also possible. Such nonbinary meters rhythmically diminish the contents of the measures they govern:

```
>>> measure = Measure((4, 10), 'c8 c c c')
>>> beam = beamtools.BeamSpanner(measure)
>>> staff = stafftools.RhythmicStaff([measure])

>>> show(staff, docs=True)
```



```
>>> note = staff.leaves[0]
>>> note.prolation
Fraction(4, 5)

>>> note.prolated_duration
Duration(1, 10)

>>> note.prolation
Fraction(4, 5)

>>> note.prolated_duration
Duration(1, 10)
```

Notes here with written duration $1/8$ carry prolation factor $4/5$ and prolated duration $1/10$.

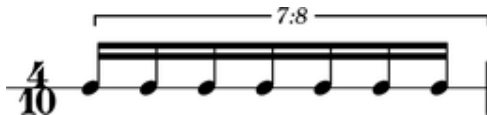
Abjad implements one of two competing nonbinary meter-interpretation schemes. The first, implicit meter-interpretation given here, follows, for example, Ferneyhough, in that nonbinary meters prolate the contents of the measures they govern implicitly, ie, without recourse to tuplet brackets. The second, explicit meter-interpretation, which we find in, for example, Sciarrino, insists instead on the presence of some tuplet bracket, usually engraved in some broken or incomplete way. The implicit meter-interpretation that Abjad implements differs from the explicit meter-interpretation native to LilyPond. Abjad will eventually implement both implicit and explicit meter-interpretation, settable on a container-by-container basis.

The prolation chain

Tuplets nest and combine freely with different types of meter. When two or more prolation donors conspire, the prolation factor they collectively bestow on leaf-level music equals the cumulative product of all prolation factors in the prolation chain. All durated components carry a prolation chain:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(4, 8), 'c16 c c c c c c')
>>> beamtools.BeamSpanner(tuplet)
BeamSpanner({c16, c16, c16, c16, c16, c16, c16})
>>> measure = Measure((4, 10), [tuplet])
>>> staff = stafftools.RhythmicStaff([measure])

>>> show(staff, docs=True)
```



```
>>> measure.multiplier
Fraction(4, 5)

>>> note = measure.leaves[0]
>>> note.prolation
Fraction(32, 35)

>>> note.prolated_duration
Duration(2, 35)
```

Notes here with written duration $1/16$ carry prolated duration $2/35$.

5.5.4 Duration types

Abjad publishes duration information about all score components.

Written duration

Abjad uses written duration to refer to the face value of notes, rests and chords prior to prolation. Abjad written duration corresponds to the informal names most frequently used when talking about note duration.

These sixteenth notes are worth a sixteenth of a whole note:

```
>>> measure = Measure((5, 16), "c16 c c c c")
>>> beam = beamtools.BeamSpanner(measure)
>>> staff = stafftools.RhythmicStaff([measure])

>>> show(staff, docs=True)
```



```
>>> note = measure[0]
>>> note.written_duration
Duration(1, 16)
```

These sixteenth notes are worth more than a sixteenth of a whole note:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(5, 16), "c8 c c c c")
>>> beam = beamtools.BeamSpanner(tuplet)
>>> measure = Measure((5, 16), [tuplet])
>>> staff = stafftools.RhythmicStaff([measure])

>>> show(staff, docs=True)
```



```
>>> note = tuplet[0]
>>> note.written_duration
Duration(1, 8)
```

The notes in these examples are ‘sixteenth notes’ that carry different prolated durations. Abjad written duration captures the fact that the note heads and flag counts of the two examples match.

Written duration is a user-assignable rational number. Users can assign and reassign the written duration of notes, rests and chords at initialization and at any time during the life of the note, rest or chord. Written durations must be assignable; see the chapter on [assignability](#) for details. Note that Abjad containers do not carry written duration.

Prolated duration

Prolation refers to the duration-scaling effects of tuplets and special types of time signature. Prolation is a way of thinking about the contribution that musical structure makes to the duration of score objects. All durated Abjad objects carry a prolated duration. Prolated duration is an emergent property of notes, tuplets and other durated objects. The prolated duration of notes, rests and chords equals the product of the written duration and prolation of those objects. The prolated duration of tuplets, measures and other containers equals the the container’s duration interface multiplied by the container’s prolation.

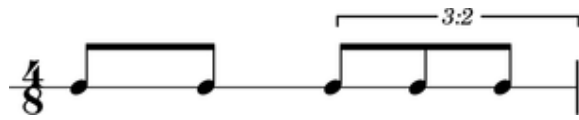
Contents duration

Abjad defines the contents duration of tuplets, measures, voices, staves and other containers equal to the sum of the preprolated duration of each of the elements in the container.

The measure here contains two eighth notes and tuplet. These elements carry preprolated durations equal to $1/8$, $1/8$ and $2/8$, respectively:

```
>>> notes = 2 * Note("c' 8")
>>> beam = beamtools.BeamSpanner(notes)
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c' 8 c c")
>>> beam = beamtools.BeamSpanner(tuplet)
>>> measure = Measure((4, 8), notes + [tuplet])
>>> staff = stafftools.RhythmicStaff([measure])

>>> show(staff, docs=True)
```



```
>>> measure.contents_duration
Duration(1, 2)
```

The contents duration of the measure here equals $1/8 + 1/8 + 2/8 = 4/8$.

Target duration

Abjad defines the target duration of fixed-duration tuplets equal to composer-settable duration to which the tuplet prolates its contents.

This fixed-duration tuplet carries a target duration equal to $4/8$:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(4, 8), "c' 8 c c c c")
>>> beam = beamtools.BeamSpanner(tuplet)
>>> measure = Measure((4, 8), [tuplet])
>>> staff = stafftools.RhythmicStaff([measure])

>>> show(staff, docs=True)
```



```
>>> tuplet.target_duration
Duration(1, 2)
```

The tuplet contents sum to $5/8$. But tuplet target duration always equals $4/8$.

Multiplied duration

Abjad defines the multiplied duration of notes, rests and chords equal to the product of written duration and leaf multiplier.

The first two notes below carry leaf multipliers equal to $2/1$:

```
>>> notes = 4 * Note("c'16")
>>> notes[0].duration_multiplier = Fraction(2, 1)
>>> notes[1].duration_multiplier = Fraction(2, 1)
>>> measure = Measure((3, 8), notes)
>>> beam = beamtools.BeamSpanner(measure)
>>> staff = stafftools.RhythmicStaff([measure])
```

```
>>> show(staff, docs=True)
```



```
>>> note = measure[0]
>>> note.written_duration
Duration(1, 16)

>>> note.duration_multiplier
Fraction(2, 1)

>>> note.written_duration * note.duration_multiplier
Duration(1, 8)
>>> note.multiplied_duration
Duration(1, 8)
```

The written duration of these first two notes equals $1/16$ and so the multiplied duration of these first two notes equals $1/16 * 2/1 = 1/8$.

5.5.5 Duration initialization

Durated Abjad classes initialize duration from arguments in the form (n, d) with numerator n and denominator d .

```
>>> note = Note("c'8.")
>>> show(note, docs=True)
```



Durated classes include notes, rests, chords, skips, tuplets and measures.

```
>>> tuplet = tuplettools.Tuplet((2, 3), "c'8 c'8 c'8")
>>> beamtools.BeamSpanner(tuplet)
>>> staff = stafftools.RhythmicStaff([tuplet])

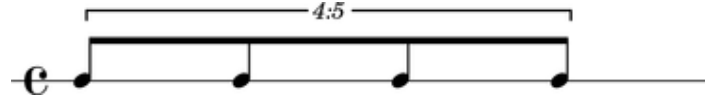
>>> show(staff, docs=True)
```



Abjad restricts notes, rests, chords and skips to durations like $3/16$ that can be written with dots, beams and flags without ties or brackets. Abjad allows arbitrary positive durations like $5/8$ for tuplets and measures.

```
>>> tuplet = tuplettools.Tuplet((5, 4), "c'8 c'8 c'8 c'8")
>>> beamtools.BeamSpanner(tuplet)
>>> staff = stafftools.RhythmicStaff([tuplet])
```

```
>>> show(staff, docs=True)
```



Abjad supports breves.

```
>>> note = Note(0, (2, 1))
```

```
>>> show(note, docs=True)
```



And longas.

```
>>> note = Note(0, (4, 1))
```

```
>>> show(note, docs=True)
```



Note: The restriction that the written durations of notes, rests, chords and skips be expressible with some combination of dots, flags and beams without recourse to ties and brackets generalizes to the condition of note_head assignability. Values (n, d) are note_head-assignable when and only when (1) d is a nonnegative integer power of 2; (2) n is either a nonnegative integer power of 2 or is a nonnegative integer power of 2, minus 1; and (3) n/d is less than or equal to 8. Condition (3) captures the fact that LilyPond provides no glyph with greater duration than the maxima (equal to eight whole notes).

Note: Integer forms like 4 as a substitute for $(4, 1)$ in `Note(0, (4, 1))` are undocumented but allowed.

Note: Abjad allows maxima note heads as in `Note(0, (8, 1))`. LilyPond implements a `\maxima` command but does not supply a corresponding glyph for the note head.

5.5.6 LilyPond multipliers

LilyPond provides an asterisk `*` operator to scale the durations of notes, rests and chords by arbitrarily positive rational values. LilyPond multipliers are invisible and generate no typographic output of their own. However, while independent from the typographic output, LilyPond multipliers do factor in in calculations of duration and time.

Abjad implements LilyPond multipliers as the settable `duration.multiplier` attribute of notes, rests and chords.

```
>>> note = Note("c'4")
>>> note.duration_multiplier = Fraction(1, 2)
>>> note.duration_multiplier
Fraction(1, 2)

>>> f(note)
c'4 * 1/2
```

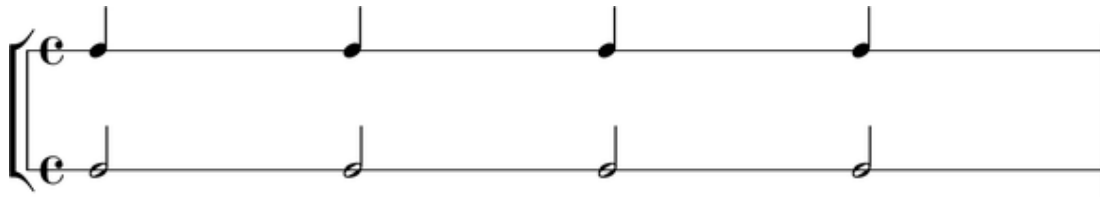
Abjad also implements a `duration.multiplied` attribute to examine the duration of a note, rest or chord as affected by the multiplier.

```
>>> note.multiplied_duration
Duration(1, 8)
```

LilyPond multipliers give the half notes here multiplied durations equal to a quarter note.

```
>>> notes = Note("c'4") * 4
>>> multiplied_note = Note(0, (1, 2))
>>> multiplied_note.duration_multiplier = Fraction(1, 2)
>>> multiplied_notes = multiplied_note * 4
>>> top = stafftools.RhythmicStaff(notes)
>>> bottom = stafftools.RhythmicStaff(multiplied_notes)
>>> staves = scoretools.StaffGroup([top, bottom])

>>> show(staves)
```



Note: Abjad models multiplication fundamentally differently than prolation . See the chapter on [Prolation](#) for more information.

Note: The LilyPond multiplication `*` operator differs from the Abjad multiplication `*` operator. LilyPond multiplication scales duration of LilyPond notes, rests and chords. Abjad multiplication copies Abjad containers and leaves.

5.5.7 Duration interfaces compared

type	core	leaf	container	measure	tuplet	fd tuplet	fm tuplet
contents	–	–	R	R	R	R	R
multiplied	–	R	–	–	–	R	R
multiplier	–	RW	–	R	R	R	RW
preprolated	R	R	R	R	R	R	R
prolated	R	R	R	R	R	R	R
prolation	R	R	R	R	R	R	R
target	–	–	–	–	–	RW	–
written	–	RW	–	–	–	–	–

The table contains a total of only four settable duration attributes, divided among only three classes. Durated Abjad classes offer up many read-only duration attributes but very few read-write duration attributes.

All classes carry all three prolation-related attributes because all classes can nest inside containers. It is possible, for example, to nest an entire voice within a fixed-duration tuplet.

Note: Leaf multipliers and tuplet multipliers differ.

5.6 Instrument marks

Instrument marks appear as markup in the left margin of your score.

5.6.1 Creating instrument marks

Use `contexttools` to create instrument marks:

```
>>> instrument_mark = contexttools.InstrumentMark('Violin ', 'Vn. ')

>>> instrument_mark
InstrumentMark('Violin ', 'Vn. ')
```

5.6.2 Attaching instrument marks to a component

Use `attach()` to attach any mark to a component:

```
>>> staff = Staff("c'4 d'4 e'4 f'4")

>>> instrument_mark.attach(staff)

>>> show(staff)
```



5.6.3 Getting the instrument mark attached to a component

Use `contexttools` to get the instrument mark attached to a component:

```
>>> contexttools.get_instrument_mark_attached_to_component(staff)
InstrumentMark('Violin ', 'Vn. ')(Staff{4})
```

5.6.4 Getting the instrument in effect for a component

Or to get the instrument currently in effect for a component:

```
>>> contexttools.get_effective_instrument(staff[1])
InstrumentMark('Violin ', 'Vn. ')(Staff{4})
```

5.6.5 Detaching instrument marks from a component one at a time

Use `detach()` to detach instrument marks from a component one at a time:

```
>>> instrument_mark.detach()

>>> instrument_mark
InstrumentMark('Violin ', 'Vn. ')
```

```
>>> show(staff)
```



5.6.6 Detaching all instrument marks attached to a component at once

Or use `contexttools` to detach instrument marks all at once:

```
>>> instrument_mark = contexttools.InstrumentMark('Violin ', 'Vn. ')
>>> instrument_mark.attach(staff)
```

```
>>> instrument_mark
InstrumentMark('Violin ', 'Vn. ') (Staff{4})
```

```
>>> show(staff)
```



```
>>> contexttools.detach_instrument_marks_attached_to_component(staff)
```

```
>>> instrument_mark
InstrumentMark('Violin ', 'Vn. ')
```

```
>>> show(staff)
```



5.6.7 Inspecting the component to which an instrument mark is attached

Use `start_component` to inspect the component to which an instrument mark is attached:

```
>>> instrument_mark = contexttools.InstrumentMark('Flute ', 'Fl. ')
>>> instrument_mark.attach(staff)
```

```
>>> show(staff)
```



```
>>> instrument_mark.start_component
Staff{4}
```

5.6.8 Inspecting the instrument name of an instrument mark

Use `instrument_name_markup` to get the instrument name of any instrument mark:

```
>>> instrument_mark.instrument_name_markup
Markup('Flute ')
```

5.6.9 Inspecting the short instrument name of an instrument mark

And use `short_instrument_name_markup` to get the short instrument name of any instrument mark:

```
>>> instrument_mark.short_instrument_name_markup
Markup('Fl. ')
```

5.7 I/O

5.7.1 Reopening Abjad PDFs

After you build a piece of notation and open with `show()` you will usually close the resulting PDF and continue working, changing your output notation in an iterative and incremental way.

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
>>> show(staff)
```

But what if you need to go back and open the resulting PDF again? Abjad provides `pdf()` for precisely this purpose. Type the following at the Abjad prompt to open the most recent PDF written by Abjad.

```
>>> pdf()
```

If you want to open not the next-to-most recent PDF generated by Abjad, pass in a `-1`. And for the next-to-next-to-most recent, pass in a `-2`, and so on.

5.7.2 Looking at LilyPond output

Abjad generates a LilyPond `.ly` file for every Abjad expression that you build and `show()`. To look at these LilyPond `.ly` files that Abjad builds behind the scenes, use `ly()`.

```
>>> ly()

% Abjad revision 2362
% 2009-06-25 10:30

\version "2.12.2"
\include "english.ly"
\include "/Users/trevorbeca/Documents/abjad/trunk/abjad/scm/abjad.scm"

\new Staff {
  c'8
  d'8
  e'8
  f'8
  g'8
  a'8
  b'8
  c''8
}
```

Abjad opens the LilyPond `.ly` file in your favorite text editor.

These LilyPond `.ly` files that Abjad generates all have the same basic structure. The current version of Abjad and the date appear first, followed by the mandatory LilyPond version string and LilyPond directives for English note names and the default Abjad `.scm` file. The remainder of the file is reserved for the LilyPond input code corresponding to the expression you just built in Abjad.

When you are done looking at the LilyPond `.ly` file quit your text editor to return to the Abjad interpreter.

5.7.3 Looking at the LilyPond log

If things go wrong when you call `show()` or one of the other Abjad functions that call LilyPond behind the scenes, it may be helpful to examine the output that LilyPond writes to the LilyPond log.

```
>>> log()

GNU LilyPond 2.12.2
Processing `1420.ly'
Parsing...
Interpreting music...
Preprocessing graphical objects...
Finding the ideal number of pages...
Fitting music on 1 page...
Drawing systems...
Layout output to `1420.ps'...
Converting to `./1420.pdf'...
```

This is the normal output that LilyPond generates every time you call the program behind. When you are done looking at the LilyPond log, quit your text editor to return to the Abjad interpreter.

5.8 LilyPond command marks

LilyPond command marks allow you to attach arbitrary LilyPond commands to Abjad score components.

5.8.1 Creating LilyPond command marks

Use `marktools` to create LilyPond command marks:

```
>>> lilypond_command_mark = marktools.LilyPondCommandMark('bar "||"', 'after')

>>> lilypond_command_mark
LilyPondCommandMark('bar "||"')
```

5.8.2 Attaching LilyPond command marks to Abjad components

Use `attach()` to attach a LilyPond command mark to any Abjad component:

```
>>> import copy
>>> staff = Staff([])
>>> key_signature = contexttools.KeySignatureMark('f', 'major')
>>> key_signature.attach(staff)
>>> staff.extend(iotools.parse_lilypond_input_string("d''16 ( c''16 fs''16 g''16 )"))
>>> staff.extend(iotools.parse_lilypond_input_string("f''16 ( e''16 d''16 c''16 )"))
```

```
>>> staff.extend(iotools.parse_lilypond_input_string("cs''16 ( d''16 f''16 d''16 )"))
>>> staff.extend(iotools.parse_lilypond_input_string("a'8 b'8"))
>>> staff.extend(iotools.parse_lilypond_input_string("d''16 ( c''16 fs''16 g''16 )"))
>>> staff.extend(iotools.parse_lilypond_input_string("f''16 ( e''16 d''16 c''16 )"))
>>> staff.extend(iotools.parse_lilypond_input_string("cs''16 ( d''16 f''16 d''16 )"))
>>> staff.extend(iotools.parse_lilypond_input_string("a'8 b'8 c''2"))

>>> lilypond_command_mark.attach(staff[-2])

>>> show(staff)
```



5.8.3 Getting the LilyPond command marks attached to an Abjad component

Use `marktools` to get the `lilypond_command_marks` attached to a leaf:

```
>>> marktools.get_lilypond_command_marks_attached_to_component(staff[-2])
(LilyPondCommandMark('bar "||"' (b'8),)
```

5.8.4 Detaching LilyPond command marks from components one at a time

Use `detach()` to detach LilyPond command marks one at a time:

```
>>> lilypond_command_mark.detach()

>>> lilypond_command_mark
LilyPondCommandMark('bar "||"')

>>> show(staff)
```



5.8.5 Detaching all LilyPond command marks attached to a component at once

Use `marktools` to detach all LilyPond command marks attached to a component at once:

```
>>> lilypond_command_mark_1 = marktools.LilyPondCommandMark('bar "||"', 'closing')
>>> lilypond_command_mark_1.attach(staff[-2])

>>> lilypond_command_mark_2 = marktools.LilyPondCommandMark('bar "||"', 'closing')
>>> lilypond_command_mark_2.attach(staff[-16])

>>> show(staff)
```



```
>>> marktools.detach_lilypond_command_marks_attached_to_component(staff[-16])
```

```
>>> show(staff)
```



5.8.6 Inspecting the component to which a LilyPond command mark is attached

Use `start_component` to inspect the component to which a LilyPond command mark is attached:

```
>>> lilypond_command_mark = marktools.LilyPondCommandMark('bar "|"', 'closing')
```

```
>>> lilypond_command_mark.attach(staff[-2])
```

```
>>> show(staff)
```



```
>>> lilypond_command_mark.start_component  
Note("b'8")
```

5.8.7 Getting and setting the command name of a LilyPond command mark

Set the `command_name` of a LilyPond command mark to change the LilyPond command a LilyPond command mark prints:

```
>>> lilypond_command_mark.command_name = 'bar "|. "'
```

```
>>> show(staff)
```



5.8.8 Copying LilyPond commands

Use `copy.copy()` to copy a LilyPond command mark:

```
>>> import copy
```

```
>>> lilypond_command_mark_copy_1 = copy.copy(lilypond_command_mark)
```

```
>>> lilypond_command_mark_copy_1  
LilyPondCommandMark('bar "|. "')
```

```
>>> lilypond_command_mark_copy_1.attach(staff[-1])
```

```
>>> show(staff)
```



Or use `copy.deepcopy()` to do the same thing.

5.8.9 Comparing LilyPond command marks

LilyPond command marks compare equal with equal command names:

```
>>> lilypond_command_mark.command_name
'bar "|".'

>>> lilypond_command_mark_copy_1.command_name
'bar "|".'

>>> lilypond_command_mark == lilypond_command_mark_copy_1
True
```

Otherwise LilyPond command marks do not compare equal.

5.9 LilyPond comments

LilyPond comments begin with the `%` sign. Abjad models LilyPond comments as marks.

5.9.1 Creating LilyPond comments

Use `marktools` to create LilyPond comments:

```
>>> comment_1 = marktools.LilyPondComment('This is a LilyPond comment before a note.', 'before')

>>> comment_1
LilyPondComment('This is a LilyPond comment before a note.')
```

5.9.2 Attaching LilyPond comments to leaves

Attach LilyPond comments to a note, rest or chord with `attach()`:

```
>>> note = Note("cs' '4")

>>> show(note, docs=True)
```



```
>>> comment_1.attach(note)

>>> f(note)
% This is a LilyPond comment before a note.
cs' '4
```

You can add LilyPond comments before, after or to the right of any leaf.

5.9.3 Attaching LilyPond comments to containers

Use `attach()` to attach LilyPond comments to a container:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> show(staff, docs=True)
```



```
>>> staff_comment_1 = marktools.LilyPondComment('Here is a LilyPond comment before the staff.', 'before')
>>> staff_comment_2 = marktools.LilyPondComment('Here is a LilyPond comment in the staff opening.', 'opening')
>>> staff_comment_3 = marktools.LilyPondComment('Here is another LilyPond comment in the staff opening.', 'opening')
>>> staff_comment_4 = marktools.LilyPondComment('LilyPond comment in the staff closing.', 'closing')
>>> staff_comment_5 = marktools.LilyPondComment('LilyPond comment after the staff.', 'after')
```

```
>>> staff_comment_1.attach(staff)
>>> staff_comment_2.attach(staff)
>>> staff_comment_3.attach(staff)
>>> staff_comment_4.attach(staff)
>>> staff_comment_5.attach(staff)
```

```
>>> f(staff)
% Here is a LilyPond comment before the staff.
\new Staff {
  % Here is a LilyPond comment in the staff opening.
  % Here is another LilyPond comment in the staff opening.
  c'8
  d'8
  e'8
  f'8
  % LilyPond comment in the staff closing.
}
% LilyPond comment after the staff.
```

You can add LilyPond comments before, after, in the opening or in the closing of any container.

5.9.4 Getting the LilyPond comments attached to a component

Use `marktools` to get all the LilyPond comments attached to a component:

```
>>> marktools.get_lilypond_comments_attached_to_component(note)
(LilyPondComment('This is a LilyPond comment before a note.')(cs''4),)
```

Abjad returns a tuple of zero or more LilyPond comments.

5.9.5 Detaching LilyPond comments from a component one at a time

Use `detach()` to detach LilyPond comments from a component one at a time:

```
>>> comment_1 = marktools.get_lilypond_comments_attached_to_component(note)[0]
```

```
>>> comment_1.detach()
LilyPondComment('This is a LilyPond comment before a note.')
```



```
>>> f(note)
cs''4
```

5.9.6 Detaching all LilyPond comments attached to a component at once

Or use `marktools` to detach all LilyPond comments attached to a component at once:

```
>>> for comment in marktools.get_lilypond_comments_attached_to_component(staff): print comment
LilyPondComment('Here is a LilyPond comment before the staff.')(Staff{4})
LilyPondComment('Here is a LilyPond comment in the staff opening.')(Staff{4})
LilyPondComment('Here is another LilyPond comment in the staff opening.')(Staff{4})
LilyPondComment('LilyPond comment in the staff closing.')(Staff{4})
LilyPondComment('LilyPond comment after the staff.')(Staff{4})

>>> marktools.detach_lilypond_comments_attached_to_component(staff)

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
```

5.9.7 Inspecting the component to which a LilyPond comment is attached

Use `start_component` to inspect the component to which a LilyPond comment is attached:

```
>>> comment_1.attach(note)

>>> comment_1.start_component
Note("cs''4")
```

5.9.8 Inspecting contents string of a LilyPond comment

Use `contents_string` to inspect the written contents of a LilyPond comment:

```
>>> comment_1.contents_string
'This is a LilyPond comment before a note.'
```

5.10 LilyPond files

5.10.1 Making LilyPond files

Make a basic LilyPond input file with the `lilyfiletools` package:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> lilypond_file = lilyfiletools.make_basic_lilypond_file(staff)

>>> lilypond_file
LilyPondFile(Staff{4})
```

5.10.2 Inspecting file output

LilyPond input files that you create this way come equipped with many attributes that appear in file output:

```
>>> f(lilypond_file)
% Abjad revision 4746
% 2011-09-04 17:36

\version "2.15.9"
\include "english.ly"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"

\score {
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
}
```

5.10.3 Setting default paper size

Set default LilyPond paper size like this:

```
>>> lilypond_file.default_paper_size = '11x17', 'landscape'

>>> f(lilypond_file)
% Abjad revision 4746
% 2011-09-04 17:36

\version "2.15.9"
\include "english.ly"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"

#(set-default-paper-size "11x17" 'landscape)

\score {
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
}
```

5.10.4 Setting global staff size

Set global staff size like this:

```
>>> lilypond_file.global_staff_size = 16

>>> f(lilypond_file)
% Abjad revision 4746
% 2011-09-04 17:36
```

```

\version "2.15.9"
\include "english.ly"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"

#(set-default-paper-size "11x17" 'landscape)
#(set-global-staff-size 16)

\score {
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
}

```

5.11 Measures

5.11.1 Understanding measures in LilyPond

In LilyPond you specify time signatures by hand and LilyPond creates measures automatically:

```

\new Staff {
  \time 3/8
  c'8
  d'8
  e'8
  d'8
  e'8
  f'8
  \time 2/4
  g'4
  e'4
  f'4
  d'4
  c'2
}

```



Here LilyPond creates five measures from two time signatures. This happens because behind-the-scenes LilyPond time-keeping tells the program when measures start and stop and how to draw the barlines that come between them.

5.11.2 Understanding measures in Abjad

Measures are optional in Abjad, too, and you may omit them in favor of time signatures:

```

>>> staff = Staff("c'8 d'8 e'8 d'8 e'8 f'8 g'4 e'4 f'4 d'4 c'2")

>>> contexttools.TimeSignatureMark((3, 8))(staff)
>>> contexttools.TimeSignatureMark((2, 4))(staff[6])

```

```
>>> show(staff)
```



But you may also include explicit measures in the Abjad scores you build. The following sections explain how.

5.11.3 Creating measures

Create a measure with a meter and music:

```
>>> measure = Measure((3, 8), "c'8 d'8 e'8")
```

```
>>> f(measure)
{
    \time 3/8
    c'8
    d'8
    e'8
}
```

```
>>> show(measure)
```



5.11.4 Working with dynamic measures

Dynamic measures adjust their time signatures on the fly as you add and remove music.

Create dynamic measures without a time signature:

```
>>> measure = measuretools.DynamicMeasure("c'8 d'8 e'8")
```

```
>>> show(measure)
```



5.11.5 Adding music to dynamic measures

Add music to dynamic measures the same as to all containers:

```
>>> measure.extend([Note("fs'8"), Note("gs'8")])
```

```
>>> show(measure)
```



5.11.6 Removing music from dynamic measures

Remove music from dynamic measures the same as with other containers:

```
>>> del (measure[1:3])
```

```
>>> show(measure)
```



5.11.7 Setting the denominator of dynamic measures

You can set the denominator of dynamic measures to any integer power of 2:

```
>>> measure.denominator = 32
```

```
>>> show(measure)
```



5.11.8 Suppressing the meter of dynamic measures

You can temporarily suppress the meter of dynamic measures:

```
>>> measure.suppress_meter = True
```

```
>>> f(measure)
```

```
{
    c' 8
    fs' 8
    gs' 8
}
```

LilyPond will engrave the last active meter.

5.11.9 Working with anonymous measures

Anonymous determine their time signatures on the fly and then hide them at format time.

Create anonymous measures without a time signature:

```
>>> measure = measuretools.AnonymousMeasure("c' 8 d' 8 e' 8")
```

```
>>> show(measure)
```



5.11.10 Adding music to anonymous measures

Add music to anonymous measures the same as to other containers:

```
>>> measure.extend([Note("fs'8"), Note("gs'8")])
```

```
>>> show(measure)
```



5.11.11 Removing music from anonymous measures

Remove music from anonymous measure the same as from other containers:

```
>>> del(measure[1:3])
```

```
>>> show(measure)
```



5.12 Notes

5.12.1 Making notes from a string

You can make notes from string:

```
>>> note = Note("c'4")
```

```
>>> show(note, docs=True)
```



5.12.2 Making notes from chromatic pitch number and duration

You can also make notes from chromatic pitch number and duration:

```
>>> note = Note(0, Duration(1, 4))
```

```
>>> show(note, docs=True)
```



(You even use `Note("c'4")` to create notes with numbers alone.)

5.12.3 Getting the written pitch of notes

You can get the written pitch of notes:

```
>>> note.written_pitch
NamedChromaticPitch("c' ")
```

5.12.4 Changing the written pitch of notes

And you can change the written pitch of notes:

```
>>> note.written_pitch = "cs' "
```



(You can use `note.written_pitch = 1` to change pitch with numbers, too.)

5.12.5 Getting the duration attributes of notes

Get the written duration of notes like this:

```
>>> note.written_duration
Duration(1, 4)
```

Which is usually the same as preprolated duration:

```
>>> note.preprolated_duration
Duration(1, 4)
```

And prolated duration:

```
>>> note.prolated_duration
Duration(1, 4)
```

Except for notes inside a tuplet:

```
>>> tuplet = Tuplet(Fraction(2, 3), [Note("c'4"), Note("d'4"), Note("e'4")])
>>> show(tuplet, docs=True)
```



```
>>> note = tuplet[0]
```

Tupletted notes carry written duration:

```
>>> note.written_duration
Duration(1, 4)
```

Prolation:

```
>>> note.prolation
Fraction(2, 3)
```

And prolated duration that is the product of the two:

```
>>> note.prolated_duration
Duration(1, 6)
```

5.12.6 Changing the written duration of notes

You can change the written duration of notes:

```
>>> tuplet[0].written_duration = Duration(1, 8)
>>> tuplet[1].written_duration = Duration(1, 8)
>>> tuplet[2].written_duration = Duration(1, 8)

>>> show(tuplet, docs=True)
```



Other duration attributes are read-only.

5.12.7 Overriding notes

The notes below are black with fixed thickness and predetermined spacing:

```
>>> staff = Staff("c'4 d'4 e'4 f'4 g'4 a'4 g'2")
>>> slur_1 = spannertools.SlurSpanner(staff[:2])
>>> slur_2 = spannertools.SlurSpanner(staff[2:4])
>>> slur_3 = spannertools.SlurSpanner(staff[4:6])
```

```
>>> f(staff)
\new Staff {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    g'2
}
```

```
>>> show(staff)
```



But you can override LilyPond grobs to change the look of notes, rests and chords:

```
>>> staff[-1].override.note_head.color = 'red'
>>> staff[-1].override.stem.color = 'red'

>>> f(staff)
\new Staff {
    c'4 (
    d'4 )
```



```

    e'4 (
    f'4 )
    g'4 (
    a'4 )
    \once \override NoteHead #'color = #red
    \once \override Stem #'color = #red
    g'2
}

>>> show(staff)

```



5.12.8 Removing note overrides

Delete grob overrides you no longer want:

```

>>> del(staff[-1].override.stem)

>>> f(staff)
\new Staff {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    \once \override NoteHead #'color = #red
    g'2
}

>>> show(staff)

```



5.13 Pitches

Named chromatic pitches are the everyday pitches attached to notes and chords:

```

>>> note = Note("cs' '8")

>>> note.written_pitch
NamedChromaticPitch("cs' ' ")

```

5.13.1 Creating pitches

Use pitch tools to create named chromatic pitches:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")

>>> named_chromatic_pitch
NamedChromaticPitch("cs'")
```

5.13.2 Inspecting the name of a pitch

Use `str()` to get the name of named chromatic pitches:

```
>>> str(named_chromatic_pitch)
cs''
```

5.13.3 Inspecting the octave of a pitch

Get the octave number of named chromatic pitches with `octave_number`:

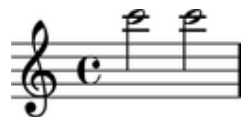
```
>>> named_chromatic_pitch.octave_number
5
```

5.13.4 Working with pitch deviation

Use deviation to model the fact that two pitches differ by a fraction of a semitone:

```
>>> note_1 = Note(24, (1, 2))
>>> note_2 = Note(24, (1, 2))
>>> staff = Staff([note_1, note_2])

>>> show(staff)
LilyPond file written to 'pitch-deviation-1.ly' ...
```



```
>>> note_2.written_pitch = pitchtools.NamedChromaticPitch(24, deviation = -31)
```

The pitch of the first note is greater than the pitch of the second:

```
>>> note_1.written_pitch > note_2.written_pitch
False
```

Use markup to include indications of pitch deviation in your score:

```
>>> markuptools.Markup(note_2.written_pitch.deviation_in_cents, Up)(note_2)
LilyPond file written to 'pitch-deviation-2.ly' ...
```



5.13.5 Sorting pitches

Named chromatic pitches sort by octave, diatonic pitch-class and accidental, in that order:

```
>>> pitchtools.NamedChromaticPitch('es') < pitchtools.NamedChromaticPitch('ff')
True
```

5.13.6 Comparing pitches

Compare named chromatic pitches to each other:

```
>>> named_chromatic_pitch_1 = pitchtools.NamedChromaticPitch("c' ' ")
>>> named_chromatic_pitch_2 = pitchtools.NamedChromaticPitch("d' ' ")

>>> named_chromatic_pitch_1 == named_chromatic_pitch_2
False

>>> named_chromatic_pitch_1 != named_chromatic_pitch_2
True

>>> named_chromatic_pitch_1 > named_chromatic_pitch_2
False

>>> named_chromatic_pitch_1 < named_chromatic_pitch_2
True

>>> named_chromatic_pitch_1 >= named_chromatic_pitch_2
False

>>> named_chromatic_pitch_1 <= named_chromatic_pitch_2
True
```

5.13.7 Converting one type of pitch to another

Convert any named chromatic pitch to a named diatonic pitch:

```
>>> named_chromatic_pitch.named_diatonic_pitch
NamedDiatonicPitch("c' ' ")
```

To a numbered chromatic pitch:

```
>>> named_chromatic_pitch.numbered_chromatic_pitch
NumberedChromaticPitch(13)
```

Or to a numbered diatonic pitch:

```
>>> named_chromatic_pitch.numbered_diatonic_pitch
NumberedDiatonicPitch(7)
```

5.13.8 Converting pitches to pitch-classes

Convert any named chromatic pitch to a named chromatic pitch-class:

```
>>> named_chromatic_pitch.named_chromatic_pitch_class
NamedChromaticPitchClass('cs')
```

To a named diatonic pitch-class:

```
>>> named_chromatic_pitch.named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

To a numbered chromatic pitch-class:

```
>>> named_chromatic_pitch.numbered_chromatic_pitch_class
NumberedChromaticPitchClass(1)
```

Or to a numbered diatonic pitch-class:

```
>>> named_chromatic_pitch.numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

5.13.9 Copying pitches

Use `copy.copy()` to copy named chromatic pitches:

```
>>> import copy

>>> copy.copy(named_chromatic_pitch)
NamedChromaticPitch("cs' ' ")
```

Or use `copy.deepcopy()` to do the same thing.

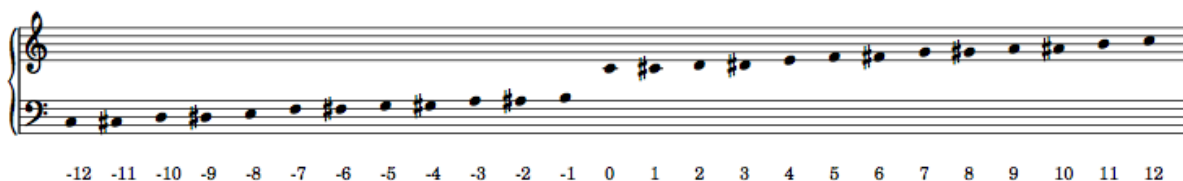
5.13.10 Accidental abbreviations

Abjad abbreviates accidentals according to the LilyPond `english.ly` module:

accidental name	abbreviation
quarter sharp	‘qs’
quarter flat	‘qf’
sharp	‘s’
flat	‘f’
three-quarters sharp	‘tqs’
three-quarters flat	‘tqf’
double sharp	‘ss’
double flat	‘ff’

5.13.11 Chromatic pitch numbers

Abjad numbers chromatic pitches by semitone with middle C set equal to 0:



The code to generate this table is as follows:

```
score, treble_staff, bass_staff = scoretools.make_empty_piano_score()
duration = Fraction(1, 32)

treble = measuretools.AnonymousMeasure([])
bass = measuretools.AnonymousMeasure([])

treble_staff.append(treble)
bass_staff.append(bass)

pitches = range(-12, 12 + 1)

pitchtools.set_default_accidental_spelling('sharps')

for i in pitches:
    note = Note(i, duration)
    rest = Rest(duration)
    clef = pitchtools.suggest_clef_for_named_chromatic_pitches([note.pitch])
    if clef == contexttools.ClefMark('treble'):
        treble.append(note)
        bass.append(rest)
    else:
        treble.append(rest)
        bass.append(note)
    diatonic_pitch_number = str(note.pitch.numbered_chromatic_pitch)
    markuptools.Markup(diatonic_pitch_number, Down)(bass[-1])

score.override.rest.transparent = True
score.override.stem.stencil = False
```

5.13.12 Diatonic pitch numbers

Abjad numbers diatonic pitches by staff space with middle C set equal to 0:



The code to generate this table is as follows:

```
score, treble_staff, bass_staff = scoretools.make_empty_piano_score()
duration = Fraction(1, 32)

treble = measuretools.AnonymousMeasure([])
bass = measuretools.AnonymousMeasure([])

treble_staff.append(treble)
bass_staff.append(bass)

pitches = []
diatonic_pitches = [0, 2, 4, 5, 7, 9, 11]

pitches.extend([-24 + x for x in diatonic_pitches])
pitches.extend([-12 + x for x in diatonic_pitches])
pitches.extend([0 + x for x in diatonic_pitches])
```

```

pitches.extend([12 + x for x in diatonic_pitches])
pitches.append(24)
pitchtools.set_default_accidental_spelling('sharps')

for i in pitches:
    note = Note(i, duration)
    rest = Rest(duration)
    clef = pitchtools.suggest_clef_for_named_chromatic_pitches([note.pitch])
    if clef == contexttools.ClefMark('treble'):
        treble.append(note)
        bass.append(rest)
    else:
        treble.append(rest)
        bass.append(note)
    diatonic_pitch_number = abs(note.pitch.numbered_diatonic_pitch)
    markuptools.Markup(diatonic_pitch_number, Down)(bass[-1])

score.override.rest.transparent = True
score.override.stem.stencil = False

```

5.13.13 Octave designation

Abjad designates octaves with both numbers and ticks:

Octave notation	Tick notation
C7	c''''
C6	c'''
C5	c''
C4	c'
C3	c
C2	c,
C1	c,,

5.13.14 Accidental spelling

Abjad chooses between enharmonic spellings at pitch-initialization according to the following table:

Chromatic pitch-class number	Chromatic pitch-class name (default)
0	C
1	C#
2	D
3	Eb
4	E
5	F
6	F#
7	G
8	Gb
9	A
10	Bb
11	B

```

>>> staff = Staff([Note(n, (1, 8)) for n in range(12)])
>>> show(staff)
LilyPond file written to 'pitch-conventions-1.ly' ...

```



Use pitch tools to respell with sharps:

```
>>> pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps(staff)
>>> show(staff)
LilyPond file written to 'pitch-conventions-2.ly' ...
```



Or flats:

```
>>> pitchtools.respell_named_chromatic_pitches_in_expr_with_flats(staff)
>>> show(staff)
LilyPond file written to 'pitch-conventions-3.ly' ...
```



5.14 Working with lists of numbers

Python provides a built-in `list` class that you can use to carry around almost anything. The examples here show how to create a list of numbers and then do things with the numbers in the list.

Create a list with square brackets.

```
>>> my_list = [23, 7, 10, 18, 13, 20, 3, 2, 18, 9, 14, 3]
>>> my_list
[23, 7, 10, 18, 13, 20, 3, 2, 18, 9, 14, 3]
```

Use `len()` to find the number of elements in any list.

```
>>> len(my_list)
12
```

Use `append()` to add one element to a list.

```
>>> my_list.append(5)
>>> my_list
[23, 7, 10, 18, 13, 20, 3, 2, 18, 9, 14, 3, 5]
```

Use `extend()` to extend one list with the contents of another.

```
>>> my_other_list = [19, 11, 4, 10, 12]
>>> my_list.extend(my_other_list)
>>> my_list
[23, 7, 10, 18, 13, 20, 3, 2, 18, 9, 14, 3, 5, 19, 11, 4, 10, 12]
```

Use `reverse()` to reverse the elements in a list.

```
>>> my_list.reverse()
>>> my_list
[12, 10, 4, 11, 19, 5, 3, 14, 9, 18, 2, 3, 20, 13, 18, 10, 7, 23]
```

You can return a single value from a list with a numeric index.

```
>>> my_list[0]
12
>>> my_list[1]
10
>>> my_list[2]
4
```

You can return many values from a list with slice notation.

```
>>> my_list[:4]
[12, 10, 4, 11]
```

More information on these and all other operations defined on the built-in Python `list` is available in the [Python tutorial](#).

5.15 Rests

5.15.1 Making rests from strings

You can make rests from a string:

```
>>> rest = Rest('r8')

>>> show(rest, docs=True)
```



5.15.2 Making rests from durations

You can also make rests from a duration:

```
>>> rest = Rest(Duration(1, 4))

>>> show(rest, docs=True)
```



(You can even use `Rest((1, 8))` to make rests from a duration pair.)

5.15.3 Getting the duration attributes of rests

Get the written duration of rests like this:

```
>>> rest.written_duration
Duration(1, 4)
```

Which is usually the same as preprolated duration:


```
>>> rest.preprolated_duration
Duration(1, 4)
```

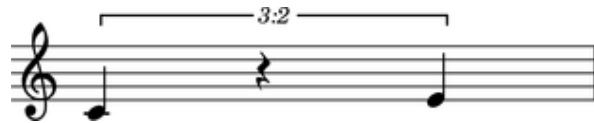
And prolated duration:

```
>>> rest.prolated_duration
Duration(1, 4)
```

Except for rests inside a tuplet:

```
>>> tuplet = Tuplet(Fraction(2, 3), [Note("c'4"), Rest('r4'), Note("e'4")])

>>> show(tuplet, docs=True)
```



```
>>> rest = tuplet[1]
```

Tupletted rests carry written duration:

```
>>> rest.written_duration
Duration(1, 4)
```

Prolation:

```
>>> rest.prolation
Fraction(2, 3)
```

And prolated duration that is the product of the two:

```
>>> rest.prolated_duration
Duration(1, 6)
```

5.15.4 Changing the written duration of rests

You can change the written duration of notes and rests:

```
>>> tuplet[0].written_duration = Duration(1, 8)
>>> tuplet[1].written_duration = Duration(1, 8)
>>> tuplet[2].written_duration = Duration(1, 8)

>>> show(tuplet, docs=True)
```



Other duration attributes are read-only.

5.16 Scores

5.16.1 Creating scores

Create a score like this:

```
>>> treble_staff_1 = Staff("e'4 d'4 e'4 f'4 g'1")
>>> treble_staff_2 = Staff("c'2. b8 a8 b1")

>>> score = Score([treble_staff_1, treble_staff_2])

>>> show(score)
```



5.16.2 Inspecting score music

Return score components with `music`:

```
>>> score.music
(Staff{5}, Staff{4})
```

5.16.3 Inspecting score length

Get score length with `len()`:

```
>>> len(score)
2
```

5.16.4 Inspecting score duration

Score contents duration is equal to the duration of the longest component in score:

```
>>> score.contents_duration
Duration(2, 1)
```

5.16.5 Adding one component to the bottom of a score

Add one component to the bottom of a score with `append`:

```
>>> bass_staff = Staff("g4 f4 e4 d4 d1")
>>> contexttools.ClefMark('bass')(bass_staff)

>>> score.append(bass_staff)

>>> show(score)
```



5.16.6 Finding the index of a score component

Find the index of a score component with `index`:

```
>>> score.index(treble_staff_1)
0
```

5.16.7 Removing a score component by index

Use `pop` to remove a score component by index:

```
>>> score.pop(1)

>>> show(score)
```



5.16.8 Removing a score component by reference

Remove a score component by reference with `remove`:

```
>>> score.remove(treble_staff_1)

>>> show(score)
```



5.16.9 Testing score containment

Use `in` to find out whether a score contains a given component:

```
>>> treble_staff_1 in score
False
```

```
>>> treble_staff_2 in score
False

>>> bass_staff in score
True
```

5.16.10 Naming scores

You can name Abjad scores:

```
>>> score.name = 'Example Score'
```

Score names appear in LilyPond input:

```
>>> f(score)
\context Score = "Example Score" <<
  \new Staff {
    \clef "bass"
    g4
    f4
    e4
    d4
    d1
  }
>>
```

But do not appear in notational output:

```
>>> show(score)
```



5.17 Spanners

5.17.1 Overriding spanners

The symbols below are black with fixed thickness and predetermined spacing:

```
>>> staff = Staff("c'4 d'4 e'4 f'4 g'4 a'4 g'2")
>>> slur_1 = spannertools.SlurSpanner(staff[:2])
>>> slur_2 = spannertools.SlurSpanner(staff[2:4])
>>> slur_3 = spannertools.SlurSpanner(staff[4:6])

>>> f(staff)
\new Staff {
  c'4 (
  d'4 )
  e'4 (
  f'4 )
  g'4 (
  a'4 )
  g'2
}
```

```
>>> show(staff)
```



But you can override LilyPond grobs to change the look of spanners:

```
>>> slur_1.override.slur.color = 'red'
>>> slur_3.override.slur.color = 'red'
```

```
>>> f(staff)
\new Staff {
  \override Slur #'color = #red
  c'4 (
    d'4 )
  \revert Slur #'color
  e'4 (
    f'4 )
  \override Slur #'color = #red
  g'4 (
    a'4 )
  \revert Slur #'color
  g'2
}
```

```
>>> show(staff)
```



5.17.2 Overriding the components to which spanners attach

You can override LilyPond grobs to change spanners' contents:

```
>>> slur_2.override.slur.color = 'blue'
>>> slur_2.override.note_head.color = 'blue'
>>> slur_2.override.stem.color = 'blue'
```

```
>>> f(staff)
\new Staff {
  \override Slur #'color = #red
  c'4 (
    d'4 )
  \revert Slur #'color
  \override NoteHead #'color = #blue
  \override Slur #'color = #blue
  \override Stem #'color = #blue
  e'4 (
    f'4 )
  \revert NoteHead #'color
  \revert Slur #'color
  \revert Stem #'color
  \override Slur #'color = #red
  g'4 (
```

```
    a'4 )
    \revert Slur #'color
    g'2
}
```

```
>>> show(staff)
```



5.17.3 Removing spanner overrides

Delete grob overrides you no longer want:

```
>>> del(slur_1.override.slur)
>>> del(slur_3.override.slur)

>>> f(staff)
\new Staff {
    c'4 (
    d'4 )
    \override NoteHead #'color = #blue
    \override Slur #'color = #blue
    \override Stem #'color = #blue
    e'4 (
    f'4 )
    \revert NoteHead #'color
    \revert Slur #'color
    \revert Stem #'color
    g'4 (
    a'4 )
    g'2
}
```

```
>>> show(staff)
```



5.18 Staves

5.18.1 Creating staves

Create staves like this:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'4 c''1")
>>> show(staff)
```



5.18.2 Inspecting staff music

Return staff components with `music`:

```
>>> staff.music
(Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'8"), Note("a'8"), Note("b'4"), Note("c'8"))
```

5.18.3 Inspecting staff length

Get staff length with `len()`:

```
>>> len(staff)
8
```

5.18.4 Inspecting staff duration

Staff contents durations equals the sum of staff components' duration:

```
>>> staff.contents_duration
Duration(2, 1)
```

5.18.5 Adding one component to the end of a staff

Add one component to the end of a staff with `append`:

```
>>> staff.append(Note("d'2"))

>>> show(staff)
```



5.18.6 Adding many components to the end of a staff

Add many components to the end of a staff with `extend`:

```
>>> notes = [Note("e'8"), Note("d'8"), Note("c'4")]
>>> staff.extend(notes)

>>> show(staff)
```



5.18.7 Finding the index of a staff component

Find staff component index with `index`:

```
>>> notes[0]
Note("e'8")
```

```
>>> staff.index(notes[0])
9
```

5.18.8 Removing a staff component by index

Use `pop` to remove a staff component by index:

```
>>> staff[8]
Note("d' '2")

>>> staff.pop(8)

>>> show(staff)
```



5.18.9 Removing a staff component by reference

Remove staff components by reference with `remove`:

```
>>> staff.remove(staff[-1])

>>> show(staff)
```



5.18.10 Naming staves

You can name Abjad staves:

```
>>> staff.name = 'Example Staff'
```

Staff names appear in LilyPond input:

```
>>> f(staff)
\context Staff = "Example Staff" {
    c' 8
    d' 8
    e' 8
    f' 8
    g' 8
    a' 8
    b' 4
    c' ' 1
    e' ' 8
    d' ' 8
}
```

But not in notational output:


```
>>> show(staff)
```



5.18.11 Forcing context

Staff context equals 'Staff' by default:

```
>>> staff.context
'Staff'
```

You can force staff context:

```
>>> staff.context = 'CustomUserStaff'
```

```
>>> staff.context
'CustomUserStaff'
```

```
>>> f(staff)
\context CustomUserStaff = "Example Staff" {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'4
    c''1
    e''8
    d''8
}
```

Force context when you have defined a new LilyPond context.

5.19 Tuplets

5.19.1 Making a tuplet from a LilyPond input string

You can make an Abjad tuplet from a multiplier and a LilyPond input string:

```
>>> tuplet = Tuplet(Fraction(2, 3), "c'8 d'8 e'8")
```

```
>>> show(tuplet)
```



5.19.2 Making a tuplet from a list of other Abjad components

You can also make a tuplet from a multiplier and a list of other Abjad components:

```
>>> leaves = [Note("fs'8"), Note("g'8"), Rest('r8')]
>>> tuplet = Tuplet(Fraction(2, 3), leaves)
>>> show(tuplet)
```



5.19.3 Understanding the interpreter display of a tuplet

The interpreter display of an Abjad tuplet contains three parts:

```
>>> tuplet
Tuplet(2/3, [fs'8, g'8, r8])
```

`Tuplet` tells you the tuplet's class.

`2/3` tells you the tuplet's multiplier.

The list `[fs'8, g'8, r8]` shows the top-level components the tuplet contains.

5.19.4 Understanding the string representation of a tuplet

The string representation of a tuplet contains four parts:

```
>>> print tuplet
{* 3:2 fs'8, g'8, r8 *}
```

Curly braces `{` and `}` indicate that the tuplet's music is interpreted sequentially instead of in parallel.

The asterisks `*` denote a fixed-multiplier tuplet.

`3:2` tells you the tuplet's ratio.

The remaining arguments show the top-level components of tuplet.

5.19.5 Inspecting the LilyPond format of a tuplet

Get the LilyPond input format of any Abjad object with `format`:

```
>>> tuplet.format
"\times 2/3 {\n\tfs'8\n\tg'8\n\tr8\n}"
```

Use `f()` as a short-cut to print the LilyPond format of any Abjad object:

```
>>> f(tuplet)
\times 2/3 {
    fs'8
    g'8
    r8
}
```

5.19.6 Inspecting the music in a tuplet

Get the music in any Abjad container with `music`:

```
>>> tuplet.music
(Note("fs'8"), Note("g'8"), Rest('r8'))
```

Abjad returns a read-only tuple of components.

5.19.7 Inspecting a tuplet's leaves

Get the leaves in any Abjad container with `leaves`:

```
>>> tuplet.leaves
(Note("fs'8"), Note("g'8"), Rest('r8'))
```

Abjad returns a read-only tuple of leaves.

5.19.8 Getting the length of a tuplet

Get the length of any Abjad container with `len()`:

```
>>> len(tuplet)
3
```

The length of every Abjad container is defined equal to the number of top-level components present in the container.

5.19.9 Getting the duration attributes of a tuplet

You set the multiplier of a tuplet at initialization:

```
>>> tuplet.multiplier
Fraction(2, 3)
```

The contents durations of a tuplet equals the sum of written durations of the components in the tuplet:

```
>>> tuplet.contents_duration
Duration(3, 8)
```

The multiplied duration of a tuplet equals the product of the tuplet's multiplier and the tuplet's contents duration:

```
>>> tuplet.multiplied_duration
Duration(1, 4)
```

5.19.10 Understanding rhythmic augmentation and diminution

A tuplet with a multiplier less than 1 constitutes a type of rhythmic diminution:

```
>>> tuplet.multiplier
Fraction(2, 3)

>>> tuplet.is_diminution
True
```

A tuplet with a multiplier greater than 1 is a type of rhythmic augmentation:

```
>>> tuplet.is_augmentation
False
```

5.19.11 Understanding binary and nonbinary tuplets

A tuplet is considered binary if the numerator of the tuplet multiplier is an integer power of 2:

```
>>> tuplet.multiplier
Fraction(2, 3)
```

```
>>> tuplet.is_binary
True
```

Other tuplets are nonbinary:

```
>>> tuplet.is_nonbinary
False
```

5.19.12 Adding one component to the end of a tuplet

Add one component to the end of a tuplet with `append`:

```
>>> tuplet.append(Note("e'4."))

>>> show(tuplet)
```



5.19.13 Adding many components to the end of a tuplet

Add many components to the end of a tuplet with `extend`:

```
>>> notes = [Note("fs'8"), Note("e'8"), Note("d'8"), Note("c'4.")]
>>> tuplet.extend(notes)

>>> show(tuplet)
```



5.19.14 Finding the index of a component in a tuplet

Find the index of a component in a tuplet with `index()`:

```
>>> notes[1]
Note("e'8")

>>> tuplet.index(notes[1])
5
```

5.19.15 Removing a tuplet component by index

Use `pop()` to remove a tuplet component by index:

```
>>> tuplet[7]
Note("c'4.")

>>> tuplet.pop(7)

>>> show(tuplet)
```



5.19.16 Removing a tuplet component by reference

Remove tuplet components by reference with `remove()`:

```
>>> tuplet.remove(tuplet[3])

>>> show(tuplet)
```



5.19.17 Overriding attributes of the LilyPond tuplet number grob

Override attributes of the LilyPond tuplet number grob like this:

```
>>> tuplet.override.tuplet_number.text = schemetools.SchemeFunction('tuplet-number::calc-fraction-text')
>>> tuplet.override.tuplet_number.color = 'red'

>>> f(tuplet)
\override TupletNumber #'color = #red
\override TupletNumber #'text = #tuplet-number::calc-fraction-text
\times 2/3 {
  fs'8
  g'8
  r8
  fs'8 [
  e'8
  d'8 ]
}
\revert TupletNumber #'color
\revert TupletNumber #'text

>>> show(tuplet)
```



See the LilyPond docs for lists of grob attributes available.

5.19.18 Overriding attributes of the LilyPond tuplet bracket grob

Override attributes of the LilyPond tuplet bracket grob like this:

```
>>> tuplet.override.tuplet_bracket.color = 'red'

>>> f(tuplet)
\override TupletBracket #'color = #red
\override TupletNumber #'color = #red
\override TupletNumber #'text = #tuplet-number::calc-fraction-text
\times 2/3 {
  fs'8
  g'8
  r8
  fs'8 [
  e'8
  d'8 ]
}
\revert TupletBracket #'color
\revert TupletNumber #'color
\revert TupletNumber #'text

>>> show(tuplet)
```



See the LilyPond docs for lists of grob attributes available.

5.20 Voices

5.20.1 Making a voice from a LilyPond input string

You can make an Abjad voice from a LilyPond input string:

```
>>> voice = Voice("c'8 d'8 e'8 f'8 g'8 a'8 b'4 c''1")

>>> show(voice)
```



5.20.2 Making a voice from a list of other Abjad components

You can also make a voice from a list of other Abjad components:

```
>>> components = [Tuplet(Fraction(2, 3), "c'4 d'4 e'4"), Note("f'2"), Note("g'1")]

>>> voice = Voice(components)

>>> show(voice)
```



5.20.3 Understanding the `repr` of a voice

The `repr` of an Abjad voice contains three parts:

```
>>> voice
Voice{3}
```

`Voice` tells you the voice's class.

`3` tells you the voice's length (which is the number of top-level components the voice contains).

Curly braces `{` and `}` tell you that the music inside the voice is interpreted sequentially rather than in parallel.

5.20.4 Inspecting the LilyPond format of a voice

Get the LilyPond input format of any Abjad object with `format`:

```
>>> voice.format
"\new Voice {\n\t\t\times 2/3 {\n\t\t\tc'4\n\t\t\td'4\n\t\t\te'4\n\t\t}\n\t\tf'2\n\t\tg'1\n}"
```

Use `f()` as a short-cut to print the LilyPond format of any Abjad object:

```
>>> f(voice)
\nnew Voice {
    \times 2/3 {
        c'4
        d'4
        e'4
    }
    f'2
    g'1
}
```

5.20.5 Inspecting the music in a voice

Get voice components with `music`:

```
>>> voice.music
(Tuplet(2/3, [c'4, d'4, e'4]), Note("f'2"), Note("g'1"))
```

Abjad returns a read-only tuple of components.

5.20.6 Inspecting a voice's leaves

Get the leaves in a voice with `leaves`:

```
>>> voice.leaves
(Note("c'4"), Note("d'4"), Note("e'4"), Note("f'2"), Note("g'1"))
```

Abjad returns a read-only tuple of leaves.

5.20.7 Getting the length of a voice

Get voice length with `len()`:

```
>>> len(voice)
3
```

The length of a voice is defined equal to the number of top-level components the voice contains.

5.20.8 Getting the duration attributes of a voice

The contents durations of a voice equals the sum of durations of the components in the voice:

```
>>> voice.contents_duration
Duration(2, 1)
```

The preprolated duration of a voice is usually equal to the voice's contents duration:

```
>>> voice.preprolated_duration
Duration(2, 1)
```

The prolated duration of a voice is usually equal to the voice's contents duration, too:

```
>>> voice.preprolated_duration
Duration(2, 1)
```

Only when you nest a very small voice inside a tuplet will the prolated and preprolated duration of a voice differ.

Voices that are not nested inside a tuplet carry a prolation of 1:

```
>>> voice.prolation
Fraction(1, 1)
```

All voice duration attributes are read-only.

5.20.9 Adding one component to the end of a voice

Add one component to the end of a voice with `append`:

```
>>> voice.append(Note("af'2"))
>>> show(voice)
```



5.20.10 Adding many components to the end of a voice

Add many components to the end of a voice with `extend`:

```
>>> notes = [Note("g'4"), Note("f'4")]
>>> voice.extend(notes)
>>> show(voice)
```




5.20.11 Finding the index of a component in a voice

Find the index of a component in a voice with `index()`:

```
>>> notes[0]
Note("g' 4")

>>> voice.index(notes[0])
4
```

5.20.12 Removing a voice component by index

Use `pop()` to remove a voice component by index:

```
>>> voice[5]
Note("f' 4")

>>> voice.pop(5)

>>> show(voice)
```



5.20.13 Removing a voice component by reference

Remove voice components by reference with `remove()`:

```
>>> voice.remove(voice[-1])

>>> show(voice)
```



5.20.14 Naming voices

You can name Abjad voices:

```
>>> voice.name = 'Upper Voice'
```

Voice names appear in LilyPond input:

```
>>> f(voice)
\context Voice = "Upper Voice" {
  \times 2/3 {
    c' 4
```

```

                d' 4
                e' 4
            }
        f' 2
        g' 1
        af' 2
    }

```

But not in notational output:

```
>>> show(voice)
```



5.20.15 Changing the context of a voice

The context of a voice is set to 'Voice' by default:

```
>>> voice.context
'Voice'
```

But you can change the context of a voice if you want:

```
>>> voice.context = 'SpeciallyDefinedVoice'
```

```
>>> voice.context
'SpeciallyDefinedVoice'
```

```
>>> f(voice)
\context SpeciallyDefinedVoice = "Upper Voice" {
    \times 2/3 {
        c' 4
        d' 4
        e' 4
    }
    f' 2
    g' 1
    af' 2
}

```

Change the context of a voice when you have defined a new LilyPond context based on a LilyPond voice.

DEVELOPER DOCUMENTATION

6.1 Codebase

6.1.1 How the Abjad codebase is laid out

The Abjad codebase comprises twelve top-level directories.

```
abjad$ ls
__init__.py  cfg          core          docs          interfaces  scr          tools
book         checks       demos         exceptions    opt         templates
```

Of these, it is in the `tools` directory that the bulk of the musical reasoning implemented in Abjad resides.

```
abjad$ ls tools/
__init__.py      importtools      markuptools      quantizationtools  stafftools
configurationtools  instrumenttools  mathtools        resttools          tempotools
chordtools       intervaltreertools  measuretools     schemetools        threadtools
componenttools   iotools          timesignaturetools  scoretools         tietools
containertools  layouttools      musicxmltools     sequencetools      tonalitytools
contexttools    leaftools        notetools         sievetools         tuplettools
durationtools   lilyfiletools    pitcharraytools  skiptools          verticalitytools
gracetools      marktools        pitchtools       spannertools       voicetools
```

The remaining sections of this chapter cover the topics necessary to familiarize developers coming to the project for the first time.

6.1.2 Removing prebuilt versions of Abjad before you check out

If you'd like to be at the cutting edge of the Abjad development then you should check out from Google Code and tell Python and your operating system about Abjad. You can do this by following the steps below.

But before you do this you should realize that there are two ways to get Abjad up and running on your computer. The first way is by downloading a compressed version of Abjad from the [Python Package Index](#). You probably did this when you first discovered Abjad and started to use the system. The second way is by following the steps below to check out a copy of the most recent version of the Abjad repository hosted on Google Code. If you already have a version of Abjad running on your computer but you haven't yet followed the steps below to check out from Google Code, then you probably downloaded a compressed version of Abjad from the Python Package Index.

Before you check out from Google Code you should remove all prebuilt versions of Abjad from your machine. The reason you need to do this is that having both a prebuilt version of Abjad and a Subversion-managed version of Abjad on your machine can confuse your operating system and lead to weird results when you try to start Abjad.

You remove prebuilt versions of Abjad resident on your computer by finding your site packages directory and removing the so-called Abjad ‘egg’ that Python has installed there. After you remove the Abjad egg from your site packages directory you will also need to remove the `abj`, `abjad` and `abjad-book` scripts from `/usr/local/bin` or from the directory that is equivalent to `/usr/local/bin` under your operating system.

First note the version of Python you’re currently running.

```
$ python --version
Python 2.6.1
```

This is important because you may have more than one version of Python installed on your machine. (Which tends especially to be the case if you’re running a Apple’s OS X.)

Then note that the site packages directory is a part of your filesystem into which Python installs third-party Python packages like Abjad. The location of the site packages directory varies from one operating system to the next and you may have to Google to find the exact location of the site packages directory on your machine. Under OS X you can check `/Library/Python/2.x/site-packages/`. Under Linux the site packages directory is usually `/usr/lib/python2.x/site-packages`.

Once you’ve found your site packages directory you can list its contents to see if Python has installed an Abjad egg in it.

```
site-packages$ ls
Abjad-2.0-py2.6.egg      Sphinx-1.0.7-py2.6.egg  py-1.3.4-py2.6.egg
Jinja2-2.5-py2.6.egg    docutils-0.7-py2.6.egg  py-1.4.0-py2.6.egg
Pygments-1.3.1-py2.6.egg easy-install.pth        py-1.4.4-py2.6.egg
README                  guppy                   pytest-2.0.0-py2.6.egg
Sphinx-1.0.1-py2.6.egg  guppy-0.1.9-py2.6.egg-info pytest-2.1.0-py2.6.egg
Sphinx-1.0.4-py2.6.egg  py-1.3.1-py2.6.egg
```

Remove any Abjad eggs Python has installed in your site packages directory.

After you’ve done this you should check `/usr/local/bin` or equivalent to see if the `abj`, `abjad` or `abjad-book` scripts are installed there.

```
bin$ ls
abj      abjad    abjad-book
```

Remove any of the three scripts you find installed there so that you can use the new versions of the scripts you will download from Google Code instead.

```
bin$ sudo rm abj*
```

Now proceed to the steps below to check out from Google Code.

6.1.3 Installing the development version

Follow the steps listed above to remove prebuilt versions of Abjad from your machine. Then follow the steps below to check out from Google Code.

1. Make sure Subversion is installed on your machine.

```
svn --version
```

If Subversion responds then it is already installed. Otherwise visit the [Subversion](http://subversion.apache.org/) website.

2. Check out a copy of the main line of the Abjad codebase.

```
svn checkout http://abjad.googlecode.com/svn/abjad/trunk abjad-trunk
```

3. Add the abjad trunk directory to your your PYTHONPATH environment variable.

```
export PYTHONPATH="/path/to/abjad-trunk:$PYTHONPATH"
```

4. Alternatively you may symlink your Python site packages directory to the abjad trunk directory.

```
ln -s /path/to/abjad-trunk /path/to/site-package/abjad
```

5. Finally, add abjad-trunk/scr/ to your PATH environment variable.

```
export PATH="/path/to/abjad-trunk/scr:$PATH"
```

You will then be able to run Abjad with the ``abjad`` command.

You now have a copy of the main line of the most recent version of the Abjad repository checked out to your machine.

6.2 Docs

The reST-based sources for the Abjad documentation are included in their entirety in every installation of Abjad. You may add to and edit these reST-based sources as soon as you install Abjad. However, to build human-readable HTML or PDF versions of the docs you will first need to download and install Sphinx.

The remaining sections of this chapter describe how the Abjad docs are laid out and how to build the docs with Sphinx.

6.2.1 How the Abjad docs are laid out

The source files for the Abjad docs are included in the `docs` directory of every Abjad install. The `docs` directory contains everything required to build HTML, PDF and other versions of the Abjad docs.

```
abjad$ ls docs/
Makefile      _templates  chapters    index.rst   scr
_static       _themes    conf.py     make.bat
```

The bulk of the Abjad docs live in `docs/chapters`. The chapter directories mirror the main sections on Abjad documentation. What you'll find as you inspect the chapter directories are a collection of `.rst` files organized into groups. The `.rst` extension identifies files written in restructured text.

One example:

```
abjad$ ls docs/chapters/appendices/glossary
index.rst
```

6.2.2 Installing Sphinx

Sphinx is the automated documentation system used by Python, Abjad and [other projects](#) implemented in Python. Because Sphinx is not included in the Python standard library you will probably need to download and install it.

First check to see if Sphinx is already installed on your machine.

```
$ sphinx-build --version
```

If Sphinx responds then the program is already installed on your machine. Otherwise visit the [Sphinx](#) website.

6.2.3 Removing old builds of the docs

After installing Sphinx, change to the Abjad `docs` directory and use the Sphinx makefile to remove any existing `docs/_build` directory prior to making a new build of the docs.

```
abjad$ cd docs

docs$ make clean
rm -rf _build/*
```

6.2.4 Generating the Abjad API

The `docs/scr` directory includes a script to generate the Abjad API. Run this script before building the Abjad docs for the first time.

```
docs$ scr/make-abjad-api
Building TOC tree ...
Now making Sphinx TOC ...

... Done.

Now building the HTML docs ...

sphinx-build -b html -d _build/doctrees . _build/html
Running Sphinx v1.0.7
loading pickled environment... done

... (many lines omitted) ...

Build finished. The HTML pages are in _build/html.
```

Rerun `make-abjad-api` any time you add or remove a public class, method or function from the codebase.

6.2.5 Building the HTML docs

Change to the Abjad `docs` directory and run `make html`.

```
abjad$ cd docs

docs$ make html
sphinx-build -b html -d _build/doctrees . _build/html
Running Sphinx v1.0.7
loading pickled environment... not found
building [html]: targets for 568 source files that are out of date
updating environment: 568 added, 0 changed, 0 removed
reading sources... [ 13%] chapters/api/debug/debugghandlertoregators
reading sources... [ 37%] chapters/api/tools/clonewp/by_leaf_counts_with_parenta
reading sources... [ 38%] chapters/api/tools/clonewp/by_leaf_range_with_parentag
reading sources... [ 38%] chapters/api/tools/componenttools/get_duration_crosser
reading sources... [ 38%] chapters/api/tools/componenttools/get_duration_preprol
reading sources... [ 39%] chapters/api/tools/componenttools/get_le_duration_prol

... (many more lines omitted) ...

writing output... [ 85%] chapters/api/tools/spannertools/give_attached_to_childr
writing output... [ 95%] chapters/fundamentals/duration/interfaces_compared/inde
```

```
writing output... [100%] index /indexdexexexng/indexxxdexindex
writing additional files... genindex modindex search
copying images... done
copying static files... done
dumping search index... done
dumping object inventory... done
build succeeded.
```

Build finished. The HTML pages are in `_build/html`.

You will then find the complete HTML version of the docs in `docs/_build/html`.

```
docs$ ls _build/
doctrees html
```

The output from Sphinx is verbose the first time you build the docs. On sequent builds, Sphinx reports changes only.

```
docs$ make html
sphinx-build -b html -d _build/doctrees . _build/html
Running Sphinx v1.0.7
loading pickled environment... done
building [html]: targets for 1 source files that are out of date
updating environment: 0 added, 1 changed, 0 removed
reading sources... [100%] chapters/devel/documentation/index
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] index ation/index
writing additional files... genindex modindex search
copying static files... done
dumping search index... done
dumping object inventory... done
build succeeded.
```

Build finished. The HTML pages are in `_build/html`.

6.2.6 Building a PDF of the docs

Building a PDF of the docs is a two-step process. First you build a LaTeX version of the docs. Then you typeset the LaTeX docs as a PDF.

First change to the Abjad docs directory.

```
abjad$ docs
```

Then make LaTeX sources of the docs.

```
docs$ make latex
sphinx-build -b latex -d _build/doctrees . _build/latex
Running Sphinx v1.0.7
loading pickled environment... done
building [latex]: all documents
updating environment: 0 added, 0 changed, 0 removed
looking for now-outdated files... none found
processing Abjad.tex... index chapters/start_here/abjad/index chapters/examples/bartok...

(... many lines omitted ...)
```

```
...ndices/pitch_conventions/images/example-3.png chapters/examples/ligeti/images/desordre.jpg
copying TeX support files... done
build succeeded.
```

Build finished; the LaTeX files are in `_build/latex`.

Run `'make all-pdf'` or `'make all-ps'` in that directory to run these through `(pdf)latex`.

Now follow the instructions provided by Sphinx and change to the LaTeX build directory.

```
docs$ cd _build/latex/
```

Then make a PDF version of the docs from the LaTeX sources.

```
latex$ make all-pdf

pdflatex 'Abjad.tex'
This is pdfTeX, Version 3.141592-1.40.3 (Web2C 7.5.6)
  %&-line parsing enabled.
entering extended mode
(/Abjad.tex
LaTeX2e <2005/12/01>
Babel <v3.8h> and hyphenation patterns for english, usenglishmax, dumylang, noh
yphenation, arabic, basque, bulgarian, coptic, welsh, czech, slovak, german, ng
erman, danish, esperanto, spanish, catalan, galician, estonian, farsi, finnish,

(... many lines omitted ...)
```

The resulting docs will appear as `Abjad.pdf` in the LaTeX build directory you're currently in.

6.2.7 Building a coverage report

Change to the Abjad docs directory and call `sphinx-build` explicitly with the coverage builder, source directory and target directory.

```
docs$ sphinx-build -b coverage . _build/coverage
Making output directory...
Running Sphinx v1.0.7
loading pickled environment... not found
building [coverage]: coverage overview
updating environment: 568 added, 0 changed, 0 removed
reading sources... [ 37%] chapters/api/tools/clonewp/by_leaf_counts_with_parenta
reading sources... [ 38%] chapters/api/tools/clonewp/by_leaf_range_with_parentag
reading sources... [ 38%] chapters/api/tools/componenttools/get_duration_crosser

... (many lines omitted) ...

reading sources... [ 85%] chapters/api/tools/spannertools/withdraw_from_containe
reading sources... [ 95%] chapters/fundamentals/duration/interfaces_compared/ind
reading sources... [100%] index t/indexdexexexng/indexxxdexindex
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
build succeeded.
```

The coverage report is now available in the `docs/_build/coverage` directory.


```
docs$ ls _build/
coverage doctrees html
```

6.2.8 Building other versions of the docs

Examine the Sphinx makefile in the Abjad `docs/` directory or change to the `docs/` directory and type `make` with no arguments to see a list of the other versions of the Abjad docs that are available to build.

```
docs$ make
Please use 'make <target>' where <target> is one of
  html          to make standalone HTML files
  dirhtml       to make HTML files named index.html in directories
  pickle        to make pickle files
  json          to make JSON files
  htmlhelp      to make HTML files and a HTML help project
  qthelp        to make HTML files and a qthelp project
  latex         to make LaTeX files, you can set PAPER=a4 or PAPER=letter
  changes       to make an overview of all changed/added/deprecated items
  linkcheck     to check all external links for integrity
  doctest       to run all doctests embedded in the documentation (if enabled)
```

6.2.9 Inserting images with `abjad-book`

Use *abjad-book* to insert snippets of notation in the docs you write in reST.

Embed Abjad code between open and close `<abjad>` `</abjad>` tags in your `.rst.raw` sourcefile and then call `abjad-book` to create a pure `.rst` file.

```
abjad-book foo.rst.raw foo.rst
```

```
Parsing file ...
Rendering "example-1.ly" ...
Rendering "example-2.ly" ...
```

You will need to build the HTML docs again to see your work.

```
make html
```

6.2.10 Updating Sphinx

It is important periodically to update your version of Sphinx. If you used `easy_install` to install Sphinx then the usual command to update Sphinx is this:

```
$ sudo easy_install -U Sphinx
```

This will usually work. But if Sphinx fails to update then it may be because you have multiple versions of Python installed on your computer. (This tends especially to be the case under Apple's OS X.)

To get around this first note the version of Python you're currently running:

```
$ python --version
Python 2.6.1
```

Then use a version-explicit form of `easy_install` to update Sphinx:

```
$ sudo easy_install-2.6 -U Sphinx
```

6.3 Tests

Abjad includes an extensive battery of tests. Abjad is in a state of rapid development and extension. Major refactoring efforts are common every six to eight months and are likely to remain so for several years. And yet Abjad continues to allow the creation of complex pieces of fully notated score in the midst of these changes. We believe this is due to the extensive coverage provided by the automated regression battery described in the following sections.

6.3.1 Automated regression?

A battery is any collection of tests. Regression tests differ from other types of test in that they are designed to be run again and again during many different stages of the development process. Regression tests help ensure that the system continues to function correctly as developers make changes to it. An automated regression battery is one that can be run automatically by some sort of driver with minimal manual intervention.

Several different test drivers are now in use in the Python community. Abjad uses `py.test`. The `py.test` distribution is not included in the Python standard library, so one of the first thing new contributors to Abjad should do is download and install `py.test`, and then run the existing battery.

6.3.2 Running the battery

Change to the directory where you have Abjad installed. Then run `py.test`.

```
abjad$ py.test
===== test session starts =====
platform darwin -- Python 2.6.1 -- pytest-2.1.0
collected 4235 items

core/LilyPondContextProxy/test/test_LilypondContextProxy__eq__.py .
core/LilyPondContextProxy/test/test_LilypondContextProxy__repr__.py .
core/LilyPondContextProxy/test/test_LilypondContextProxy__setattr__.py ..

... (many lines omitted) ...

tools/voicetools/test/test_voicetools_iterate_semantic_voices_in_expr.py .
tools/voicetools/test/test_voicetools_iterate_voices_backward_in_expr.py .
tools/voicetools/test/test_voicetools_iterate_voices_in_expr.py .

===== 4235 passed in 127.06 seconds =====
```

Abjad r4629 includes 4235 tests.

6.3.3 Reading test output

`py.test` crawls the entire directory structure from which you call it, running tests in alphabetical order. `py.test` prints the total number of tests per file in square brackets and prints test results as a single `.` dot for success or else an `F` for failure.

6.3.4 Writing tests

Project check-in standards ask that tests accompany all code committed to the Abjad repository. If you add a new function, class or method to Abjad, you should add a new test file for that function, class or method. If you fix or extend an existing function, class or method, you should find the existing test file that covers that code and then either add a completely new test to the test file or else update an existing test already present in the test file.

6.3.5 Test files start with `test_`

When `py.test` first starts up it crawls the entire directory structure from which you call it prior to running a single test. As `py.test` executes this preflight work, it looks for any files beginning or ending with the string `test` and then collects and alphabetizes these. Only after making such a catalog of tests does `py.test` begin execution. This collect-and-cache behavior leads to the important point about naming, below.

6.3.6 Avoiding name conflicts

Note that the names of **test functions** must be absolutely unique across the entire directory structure on which you call `py.test`. You must never share names between test functions. For example, you must not have two tests named `test_grob_handling_01()` **even if both tests live in different test files**. That is, a test named `test_grob_handling_01()` living in the file `test_accidental_grob_handling.py` and a second test named `test_grob_handling_01()` living in the file `test_notehead_grob_handling.py` will conflict with the each other when `py.test` runs. And, unfortunately, **“`py.test` is silent about such conflicts when it runs**. That is, should you run `py.test` with the duplicate naming situation described here, what will happen is that `py.test` will correctly run and report results for the **first** such test it finds. However, when `py.test` encounters the second like-named test, `py.test` will incorrectly report cached results for the **first** test rather than the second. The take-away is to include some sort of namespacing indicators in every test name and not to be afraid of long test names. The `test_grob_handling_01()` example given here fixes easily when the two tests rename to `test_accidental_grob_handling_01()` and `test_notehead_grob_handling_01()`.

6.3.7 Updating `py.test`

It is important periodically to update `py.test`.

The usual command to do this is:

```
$ sudo easy_install -U pytest
```

Note that `pytest` is here spelled without the intervening period.

6.3.8 Running `doctest` on the `tools` directory

The Python standard library includes the `doctest` module as way of checking the correctness of examples included in Python docstrings. The module searches for instances of the Python interpreter prompt `'>>>'` and executes any code that follows. Abjad docs display the Abjad prompt `'abjad>'` instead of the Python prompt. This means that all instances of the Abjad prompt must be changed to Python prompts before running `doctest` on the Abjad codebase. Three scripts in `abjad/scr/devel` help do this.

First change to the subdirectory of the Abjad source tree on which you'd like to run `doctest`. Then run these scripts:

```
replace-abjad-prompts-with-python-prompts
```

```
run-doctest-on-all-modules-in-tree
```

```
replace-python-prompts-with-abjad-prompts
```

After running `run-doctest-on-all-modules-in-tree` you can inspect the results that come back from `doctest` and make any fixes as required.

6.4 Scripts

The `abjad/scr/devel` directory contains scripts for Abjad developers. Add `abjad/scr/devel` to your `PATH` to use the scripts described below.

```
abjad$ ls scr/devel
abj-grep                find-multifunction-modules
abj-grp                 find-multiline-import-statements
abj-rmpycs              find-nonalphabetized-module-headers
abj-src-grp             find-nontrivial-subdirectories
abj-test-grp            find-public-helpers-without-docstrings
abj-update              find-undocumented-tools
capitalize-test-file-names
                        fix-nonalphabetized-module-headers
conjoin-multiline-import-statements
                        fix-test-case-block-comments
count-source-lines      fix-test-case-names
count-tools              fix-test-case-numbers
duplicate-test-file      format-lilypond-context-names-with-underscores
find-and-fix-manual-class-package-initializers
                        list-private-modules
find-duplicate-module-names
                        rebuild-docs
find-duplicate-tool-module-names
                        reindent-3-spaces-as-4
find-import-as-statements
                        reindent-4-spaces-as-3
find-local-import-statements
                        reindent-spaces-variably
find-lower-camel-case-definitions
                        remove-tmp-out-directories
find-lower-camel-case-modules
                        rename-public-helper
find-manual-class-loads-in-initializers
                        replace-abjad-prompts-with-python-prompts
find-misnamed-private-modules
                        replace-in-files
find-missing-test-modules
                        replace-python-prompts-with-abjad-prompts
find-module-headers      run-doctest-on-all-modules-in-tree
find-modules-with-chevrons
```

6.4.1 Searching the Abjad codebase with `abj-grep`

Abjad provides a wrapper around UNIX `grep` in the form of `abj-grep`. Use this script to recursively search the entire Abjad codebase, leaving out non-human-readable files, files located in special `.svn` Subversion subdirectories, and all files in the `abjad/documentation` directories. You can run `abj-grep` from any directory on your system; you needn't be in the Abjad source directories when you call `abj-grep`.

```
$ abj-grep 'is_assignable('
leaf/duration.py:111:         if not durationtools.is_assignable(rational):
tempo/indication.py:67:         assert durationtools.is_assignable(arg)
tools/check/are_scalable.py:12:         if not durationtools.is_assignable(candidate_duration):
tools/durationtools/is_assignable.py:5: def is_assignable(duration):
tools/durationtools/prolated_to_written.py:2: from abjad.tools.durationtools.is_assignable import is_
tools/durationtools/prolated_to_written.py:15:     if is_assignable(prolated_duration):
tools/tietools/duration_change.py:28:     if durationtools.is_assignable(new_written_duration):
tools/tuplettools/contents_scale.py:30:     if durationtools.is_assignable(multiplier):
```

6.4.2 Removing old *.pyc files with abj-rmpycs

See the section on `abj-update` below for the reasons that it is a good idea to periodically remove the byte-compiled *.pyc files that Python generates for its own use behind the scenes. Abjad supplies `abj-rmpycs` to delete all the *.pyc in the Abjad codebase, leaving other *.pyc on your system untouched.

6.4.3 Updating your development copy of Abjad with abj-update

The normal way of updating your working copy of a Subversion repository is with the `svn update` or `svn up` command. You can update your working copy of Abjad in the usual way with `svn up`. But Abjad supplies an `abj-update` script as a wrapper around the usual Subversion update commands. In addition to updating your working copy of Abjad, `abj-update` populates the `abjad/.version` file with the most recent revision number of the system, and then removes all *.pyc files from your Abjad install. The benefits here are twofold. First, Abjad adds the most recent revision number of the system to all .ly files that you generate when working with Abjad. If you do not update the Abjad version file on a regular basis, the headers in your Abjad-generated .ly files will list the wrong version of the system. Second, as is the case in working with any substantial Python codebase, it is a good idea to periodically remove the byte-compiled *.pyc files that Python creates for its own use. The reason for this is inadvertent name aliasing. That is, if there was previously a module named `foo.py` somewhere in the system and if Python had at some point imported the module and created `foo.pyc` as a byproduct, this .pyc file will remain on the filesystem even if you later decide to remove, or rename, the source `foo.py` module. This lead to confusion because days or weeks after `foo.py` has been removed, Python will still find `foo.pyc` and seem to make the contents of `foo.py` available from beyond the grave. Updating with `abj-update` takes care of these two situations.

6.4.4 Counting lines of code with count-source-lines

Run `count-source-lines` for a count of lines of count divided between source and test files.

```
abjad$ count-source-lines

source_modules: 1703
test_modules:   1812

source_lines:   73942
test_lines:     76636

total lines:    150578
test-to-source ratio is 1 : 1
```

The script is directory-dependent so you can run it any the entire Abjad codebase or any subdirectory of the codebase.

6.4.5 Global search-and-replace with replace-in-files

You probably won't need to use `replace-in-files` very often. But if you are making changes to Abjad that will cause some name, such as `FooBar`, to be globally changed everywhere in the Abjad codebase to, say to `foo_bar`, then you can use `replace-in-files` to save lots of time.

```
$ replace-in-files --help

Usage:

replace-in-files DIR OLD_TEXT NEW_TEXT [CONFIRM=true/false]

Crawl directory DIR and read every file in it recursively.
```

```
Replace OLD_TEXT with NEW_TEXT in each file.
```

```
Set CONFIRM to `false` to replace without prompting.
```

6.4.6 Adding new development scripts

If you write and then find yourself using a certain script over and over again when you're developing new code for Abjad, consider contributing back to the project so we can include your script in the next public release of Abjad. Scripts in the the Abjad script directories end with no file extension and try to be as OS-portable as possible, which usually means writing the script in Python, rather than your operating system's shell, and relying heavily on Python's `os` module.

6.5 Using abjad-book

`abjad-book` is an independent application included in every installation of Abjad. `abjad-book` allows you to write Abjad code in the middle of documents written in HTML, LaTeX or ReST. We created `abjad-book` to help us document Abjad. Our work on `abjad-book` was inspired by `lilypond-book`, which does for LilyPond much what `abjad-book` does for Abjad.

6.5.1 HTML with embedded Abjad

To see `abjad-book` in action, open a file and write some HTML by hand. Add some Abjad code to your HTML between open and close `<abjad>` `</abjad>` tags.

```
<html>

<p>This is an <b>HTML</b> document.</p>

<p>The code is standard hypertext mark-up.</p>

<p>Here is some music notation generated automatically by Abjad:</p>

<abjad>
v = Voice("c'8 d' e' f' g' a' b' c'")
beam = beamtools.BeamSpanner(v)
show(v)
</abjad>

<p>And here is more ordinary <b>HTML</b>.</p>

</html>
```

Save your the file with the name `example.html.raw`. You now have an HTML file with embedded Abjad code.

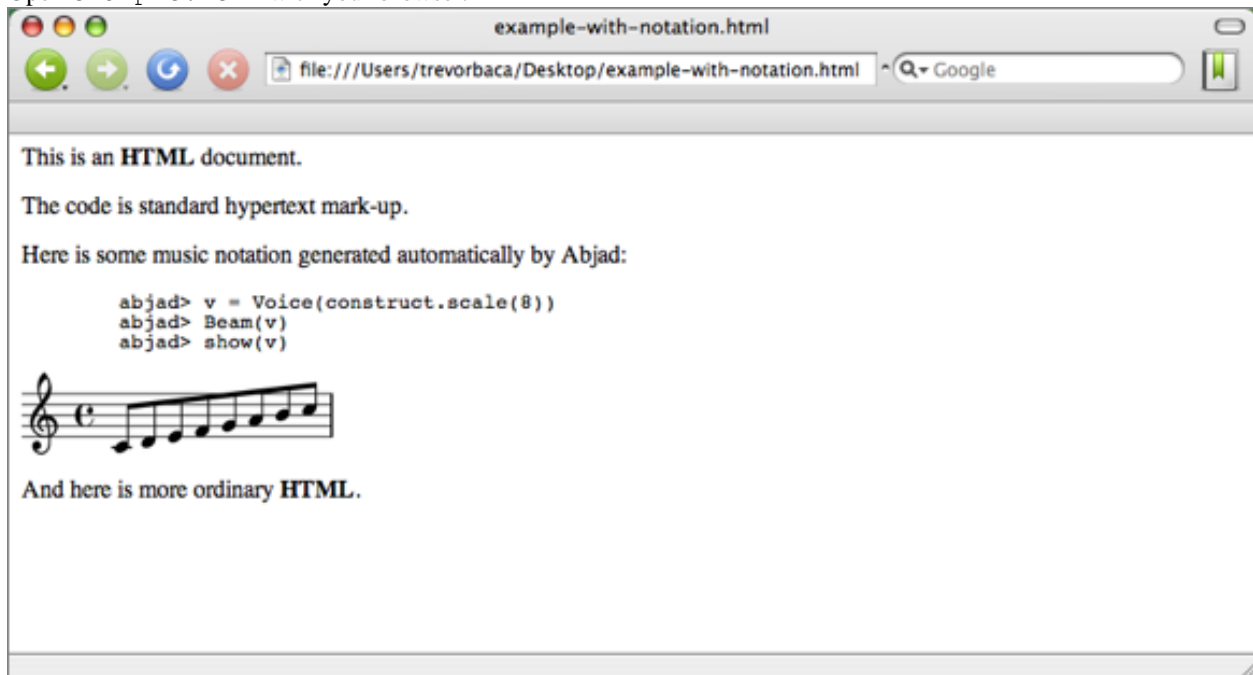
In the terminal, call `abjad-book` on `example.html.raw`.

```
$ abjad-book example.html.raw example.html
```

```
Parsing file...
Rendering "abjad-book-1.ly"...
```

The application opens `example.html.raw`, finds all Abjad code between `<abjad>` `</abjad>` tags, executes it, and then creates and inserts image files of music notation accordingly.

Open `example.html` with your browser.



That's all there is to it. `abjad-book` lets you open a file and type HTML by hand with Abjad sandwiched between the special `<abjad>` `</abjad>` tags described here. Run `abjad-book` on such a hybrid file to create pure HTML with images of music notation created by Abjad.

Note: `abjad-book` makes use of ImageMagick's `convert` application to crop and scale PNG images generated for HTML and ReST documents. For LaTeX documents, `abjad-book` uses `pdfcrop` for cropping PDFs.

6.5.2 LaTeX with embedded Abjad

You can use `abjad-book` to insert Abjad code and score excerpts into any LaTeX you create. Type the sample code below into a file.

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{listings}
\begin{document}
```

This is a standard LaTeX document with embedded Abjad.

The code below creates an Abjad measure and then prints the measure format string.

```
<abjad>
measure = Measure((5, 8), "c'8 d'8 e'8 f'8 g'8")
f(measure)
</abjad>
```

This next bit of code knows about the measure we defined earlier.

```
<abjad>
iotools.write_expr_to_ly(measure, 'abjad-book-1', docs=True) <hide
```

```
</abjad>
```

And this is the end of the our sample LaTeX document.

```
\end{document}
```

Save your file with the name `example.tex.raw`. You now have a LaTeX file with embedded Abjad code.

In the terminal, call `abjad-book` on `example.tex.raw`.

```
$ abjad-book example.tex.raw example.tex
```

```
Processing 'example.tex.raw'. Will write output to 'example.tex'...
```

```
Parsing file...
```

```
Rendering "abjad-book-1.ly"...
```

The application open `example.tex.raw`, finds all code between Abjad tags, executes it, and then creates and inserts Abjad interpreter output and PDF files of music notation. You can view the contents of the next LaTeX file `abjad-book` has created.

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{listings}
\begin{document}
```

This is a standard LaTeX document with embedded Abjad.

The code below creates an Abjad measure and then prints the measure format string.

```
\begin{lstlisting}[basicstyle=\footnotesize, tabsize=4, showtabs=false, showspace=false]
>>> measure = Measure((5, 8), "c'8 d'8 e'8 f'8 g'8")
>>> f(measure)
{
  \time 5/8
  c'8
  d'8
  e'8
  f'8
  g'8
}
\end{lstlisting}
```

This next bit of code knows about the measure we defined earlier. This code renders the measure as a PDF using a template suitable for inclusion in LaTeX documents.

```
\includegraphics{images/abjad-book-1.pdf}
```

And this is the end of the our sample LaTeX document.

```
\end{document}
```

You can now process the file `example.tex` just like any other LaTeX file, using `pdflatex` or `TexShop` or whatever LaTeX compilation program you normally use on your computer.

```
$ pdflatex example.tex
```

```
This is pdfTeXk, Version 3.141592-1.40.3 (Web2C 7.5.6)
```



```
%&-line parsing enabled.
entering extended mode
...
```

And then open the resulting PDF.

6.5.3 Using abjad-book on ReST documents

You can call abjad-book on ReST documents, too. Follow the examples given here for HTML and LaTeX documents and modify accordingly.

6.5.4 Using [hide = True]

You can add [hide = True] to any abjad-book example to show only music notation.

```
<abjad>[hide = True]
staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b''8")
iotools.write_expr_to_ly(staff, 'staff-example', docs=True)
</abjad>
```

6.6 Timing code

You can time code with Python's built-in timeit module:

```
from abjad import *
import timeit

timer = timeit.Timer('Note(0, (1, 4))', 'from __main__ import Note')
print timer.timeit(1000)

0.225436925888
```

These results show that 1000 notes take 0.23 seconds to create.

Other Python timing modules are available for download on the public Internet.

6.7 Profiling code

Profile code with profile_expr() in the iotools package:

```
>>> iotools.profile_expr('Note(0, (1, 4))')
Sun Aug 14 16:50:36 2011    _tmp_abj_profile

    327 function calls (312 primitive calls) in 0.001 CPU seconds

Ordered by: cumulative time
List reduced from 96 to 12 due to restriction <12>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      1   0.000   0.000   0.001   0.001 <string>:1(<module>)
      1   0.000   0.000   0.001   0.001 Note.py:18(__init__)
      1   0.000   0.000   0.001   0.001 Note.py:133(fset)
```

```

1      0.000      0.000      0.001      0.001 NoteHead.py:18(__init__)
1      0.000      0.000      0.001      0.001 NoteHead.py:121(fset)
1      0.000      0.000      0.001      0.001 NamedChromaticPitch.py:28(__new__)
1      0.000      0.000      0.000      0.000 Leaf.py:18(__init__)
1      0.000      0.000      0.000      0.000 chromatic_pitch_name_to_diatonic_pitch_numbe
1      0.000      0.000      0.000      0.000 octave_tick_string_to_octave_number.py:4(oct
1      0.000      0.000      0.000      0.000 re.py:134(match)
1      0.000      0.000      0.000      0.000 re.py:227(_compile)
1      0.000      0.000      0.000      0.000 sre_compile.py:501(compile)

```

These results show 327 function calls to create a note.

The `profile_expr()` function wraps the Python `cProfile` and `pstats` modules.

6.8 Memory consumption

You can examine memory consumption with tools included in the `guppy` module:

```

from guppy import hpy
hp = hpy()
hp.setrelheap()
notes = [Note(0, (1, 4)) for x in range(1000)]
h = hp.heap()
print h

```

Partition of a set of 11024 objects. Total size = 586364 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	9	124000	21	124000 21 abjad.tools.notetools.Note.Note.Note
1	1004	9	116464	20	240464 41 __builtin__.set
2	2003	18	76300	13	316764 54 list
3	1000	9	52000	9	368764 63 abjad.tools.pitchtools.NamedChromaticPitch.NamedChromat icPitch.NamedChromaticPitch
4	1000	9	44000	8	412764 70 abjad.interfaces._OffsetInterface._OffsetInterface._Off setInterface
5	1000	9	44000	8	456764 78 abjad.tools.notetools.NoteHead.NoteHead.NoteHead
6	1000	9	40000	7	496764 85 0x23add0
7	1000	9	32000	5	528764 90 abjad.interfaces.ParentageInterface.ParentageInterface. ParentageInterface
8	1011	9	28568	5	557332 95 str
9	1000	9	28000	5	585332 100 abjad.interfaces._NavigationInterface._NavigationInterf ace._NavigationInterface

<6 more rows. Type e.g. `'_.more'` to view.>

These results show 586K for 1000 notes.

You must download `guppy` from the public Internet because the module is not included in the Python standard library.

6.9 Class attributes

Consider the definition of this class:

```
class FooWithInstanceAttribute(object):

    def __init__(self):
        self.constants = (
            'red', 'orange', 'yellow', 'green',
            'blue', 'indigo', 'violet',
        )
```

1000 objects consume 176k:

```
from guppy import hpy
hp = hpy()
hp.setrelheap()
objects = [FooWithInstanceAttribute() for x in range(1000)]
h = hp.heap()
print h
```

Partition of a set of 2004 objects. Total size = 176536 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	50	140000	79	140000 79 dict of __main__.FooWithInstanceAttribute
1	1000	50	32000	18	172000 97 __main__.FooWithInstanceAttribute
2	1	0	4132	2	176132 100 list
3	1	0	348	0	176480 100 types.FrameType
4	1	0	44	0	176524 100 __builtin__.weakref
5	1	0	12	0	176536 100 int

But consider the definition of this class:

```
class FooWithSharedClassAttribute(object):

    def __init__(self):
        pass

    self.constants = (
        'red', 'orange', 'yellow', 'green',
        'blue', 'indigo', 'violet',
    )
```

1000 objects consume only 36k:

```
from guppy import hpy
hp = hpy()
hp.setrelheap()
objects = [FooWithClassAttribute() for x in range(1000)]
h = hp.heap()
print h
```

Partition of a set of 1004 objects. Total size = 36536 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	100	32000	88	32000 88 __main__.FooWithClassAttribute
1	1	0	4132	11	36132 99 list
2	1	0	348	1	36480 100 types.FrameType
3	1	0	44	0	36524 100 __builtin__.weakref
4	1	0	12	0	36536 100 int

Objects that share class attributes between them can consume less memory than objects that don't. But consider the usual provisions between class attributes and instance attributes when implementing custom classes. Class attributes make sense when objects will never modify the attribute in question. Class attributes also make sense when objects will modify the attribute in question and will desire to change the attribute in question for all other like objects at the

same time. Probably best to use instance attributes in most other cases.

6.10 Using slots

Consider the definition of this class:

```
class Foo(object)

    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
```

1000 objects consume 176k:

```
from guppy import hpy
hp = hpy()
hp.setrelheap()
objects = [Foo(1, 2, 3) for x in range(1000)]
h = hp.heap()
print h
```

Partition of a set of 2004 objects. Total size = 176536 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	50	140000	79	140000 79 dict of __main__.FooWithInstanceAttribute
1	1000	50	32000	18	172000 97 __main__.FooWithInstanceAttribute
2	1	0	4132	2	176132 100 list
3	1	0	348	0	176480 100 types.FrameType
4	1	0	44	0	176524 100 __builtin__.weakref
5	1	0	12	0	176536 100 int

But consider the definition of this class:

```
class FooWithSlots(object):

    __slots__ = ('a', 'b', 'c')
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
```

1000 objects consume only 40k:

```
from guppy import hpy
hp = hpy()
hp.setrelheap()
objects = [FooWithSlots(1, 2, 3) for x in range(1000)]
h = hp.heap()
print h
```

Partition of a set of 1004 objects. Total size = 40536 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	100	36000	89	36000 89 __main__.Bar
1	1	0	4132	10	40132 99 list
2	1	0	348	1	40480 100 types.FrameType
3	1	0	44	0	40524 100 __builtin__.weakref
4	1	0	12	0	40536 100 int

The example here confirms the Python Reference Manual 3.4.2.4: “By default, instances of both old and new-style classes have a dictionary for attribute storage. This wastes space for objects having very few instance variables. The space consumption can become acute when creating large numbers of instances.”

6.11 Coding standards

Indent with spaces, not with tabs. Use four spaces at a time:

```
def foo(x, y):  
    return x + y
```

Introduce comments with one pound sign and a single space:

```
# comment before foo  
def foo(x, y):  
    return x + y
```

Avoid `from`. Instead of `from fractions import Fraction` use:

```
import fractions
```

Favor early imports at the head of each module. Only one `import` per line.

Arrange standard library imports alphabetically at the head of each module:

```
import fractions  
import types
```

Follow standard library imports with intrapackage Abjad imports arranged alphabetically:

```
import footools  
import bartools  
import blahtools
```

Include two blank lines after `import` statements before the rest of the module:

```
import fractions  
import types  
import footools  
import bartools  
import blahtools
```

```
class Foo(object):  
    ...  
    ...
```

Wrap docstrings with triple apostrophes and align like this:

```
def foo(x, y):  
    '''This is the first line of the foo docstring.  
       This is the second line of the foo docstring.  
       And this is the last line of the foo docstring.  
    '''
```

Use paired apostrophes to delimit strings:

```
s = 'foo'
```

Use paired quotation marks to delimit strings within a string:

```
s = 'foo and "bar"'
```

Name classes in upper camelcase:

```
def FooBar(object):  
    ...  
    ...
```

Name bound methods in underscore-delimited lowercase:

```
def Foo(object):  
  
    def bar_blah(self):  
        ...  
  
    def bar_baz(self):  
        ...
```

Name module-level functions in underscore-delimited lowercase:

```
def foo_bar():  
    ...  
  
def foo_blah():  
    ...
```

Separate bound method definitions with a single empty line:

```
class FooBar(object):  
  
    def __init__(self, x, y):  
        ...  
  
    def bar_blah(self):  
        ...  
  
    def bar_baz(self):  
        ...
```

Organize the definitions of core classes into the nine following major sections:

```
class FooBar(object):  
  
    ### CLASS ATTRIBUTES ###  
  
    special_enumeration = ('foo', 'bar', 'blah')  
  
    ### INITIALIZER ###  
  
    def __init__(self, x, y):  
        ...  
  
    ### SPECIAL METHODS ###  
  
    def __repr__(self):  
        ...  
  
    def __str__(self):  
        ...
```

```

### READ-ONLY PRIVATE PROPERTIES ###

@property
def _foo(self):
    ...

### READ / WRITE PRIVATE PROPERTIES ###

@apply
def _bar():
    def fget(self):
        ...
    def fset(self, expr):
        ...
    return property(**locals())

### PRIVATE METHODS ###

def _blah(self, x, y):
    ...

### READ-ONLY PUBLIC PROPERTIES ###

@property
def foo(self):
    ...

### READ / WRITE PUBLIC PROPERTIES ###

@apply
def bar():
    def fget(self):
        ...
    def fset(self, expr):
        ...
    return property(**locals())

### PUBLIC METHODS ###

def blah(self, expr):
    ...

```

Use < less-than signs in preference to greater-than signs:

```

if x < y < z:
    ...

```

Precede private class attributes with a single underscore.

Alphabetize method names.

Alphabetize keyword arguments.

Include keyword argument names explicitly in function calls.

Limit docstring lines to 99 characters.

Limit source lines to 110 characters and use \ to break lines where necessary.

Eliminate trivial slice indices. Use `s[:4]` instead of `s[0:4]`.

Prefer new-style string formatting to old-style string interpolation. Use `'string {}'.format(expr)` instead of `'string %s content' % expr`.

Prefer list comprehensions to `filter()`, `map()` and `apply()`.

Do not abbreviate variable names. (But use `expr` for 'expression' and use `i` or `j` for loop counters.)

Name variables that represent a list or other collection of objects in the plural.

Name functions beginning with a verb. (But use `noun_to_noun` for conversion functions and `mathtools.noun` for some `mathtools` functions.)

Avoid private classes.

Avoid private functions. (But use private class methods as necessary.)

Implement package-level functions in preference to `staticmethod` class methods.

Implement only one statement per line of code.

Implement only one class per module.

Implement only one function per module.

Author one `py.test` test file for every module-level function.

Author one `py.test` test file for every bound method in the public interface of a class.

APPENDICES

7.1 From Trevor and Víctor

We are composers [Trevor Bača](#) and [Víctor Adán](#), creators of Abjad, and our earliest collaborative work dates back to shared undergraduate years in Austin. It was the mid- to late-90s and we found ourselves interested in ways of building up ever larger sets of musical materials in our scores, with ever greater amounts of musical information.

Our work then began with pitch formalization, creating materials in C and then writing the results as MIDI to hear what we'd created. Turns out that this is a fairly common gateway into materials generation for many composers, and so it was for us. Probably this was, and is, due to the ever present availability of MIDI and, to a lesser extent, CSound. But even back then it was clear to us to finding ways to embody other aspects of the musical score – from nested rhythms to the different approaches to the musical measure to the arbitrarily complex structures possible with overlapping musical voices – would require a wholly different level of consideration, and different development techniques as well.

As an example, consider flat lists of floating-point values. This basic data structure, together with the constant need some type of quantification or rounding, feeds much of most composers' work with CSound, pd and the like. It is a good thing, therefore, that essentially all modern programming languages include tools for manipulating flat lists of floats out of the box, or in the standard library. But what happens when you want to think of pitch as something much more than integers for core values with, perhaps, floats for microtones? What if you want to work with pitches as fully-fledged objects? Objects capable of carrying arbitrarily large sets of attributes and values? Objects that might group together, first into sets, and then into larger assemblages, and then into still larger complexes of pitch information loaded, or even overloaded, with cross-relationships or textural implications? Carrying this surplus of information about pitch, or the potential uses of pitch, in data structures limited to, or centered around, the list-of-floats paradigm then becomes a burden.

And what of working with rhythms not only as offset values, as implied by the list-of-floats approach, but as arbitrarily nested, stretched, compressed and stacked sets of values, as allowed by the tupleting and measure structures of conventional score? A different approach is needed.

There was, and still is, no reason to believe that general purpose programming languages and development tools should come readily supplied with the objects and methods most suitable for composerly applications. And this means that the attributes of a domain-specific language that will best meet the needs of composers interested in working formally with the full complement of capabilities in traditional score remains an open question.

We continued our work in score formalization independently until 2005, Trevor in a system that would come to be called Lascaux, and Víctor in a system dubbed Cuepatlahto. We experimented with C, Mathematica and Matlab as the core programming languages driving our systems before settling independently on Python, Víctor out of experience at MIT, where he was working on his masters at the Media Lab with Berry Vercoe, and Trevor out of the working necessities of a professional developer and engineer.

We passed through independent experiences using Finale, Sibelius, Leland Smith's SCORE, and even Adobe Illustrator as the notational rendering engines for Lascaux and Cuepatlahto. Through all of this, both systems were designed to tackle a shared set of problems. These included:

1. The difficulty involved in transcribing larger scale and highly parameterized gestures and textures into traditional Western notation.
2. The general inflexibility of closed, commercial music notation software packages.
3. The relative inability of objects on the printed page in conventional score to point to each other — or, indeed, to other objects or ideas outside the printed page — in ways rich enough to help capture, model and develop long-range, nonlocal relationships throughout our scores.

After collaborating on a joint paper describing the two systems, and after discussing collaborative design and implementation at length, both online and in weekends’ long review of our respective codebases, we decided to combine our efforts into a single, unified project. That project is now Abjad.

In our work on Abjad we strive to develop a powerful and flexible symbolic system. We picked the phrase ‘formalized score control’, or FSC, as a nod to Xenakis, who was so far ahead in so many ways, and also to highlight our primary project goal: to bring the full power of modern programming languages, and tools in mathematics, text processing, pattern recognition, and modular, iterative and incremental development to bear on all parts of the compositional process.

7.2 Why MIDI is not enough

Given that Abjad models written musical score, it might seem odd for MIDI to be even mentioned in this manual. Yet, until fairly recently, MIDI has played a role (sometimes tangential, other times fundamental) in a variety of software tools related to music notation and engraving.

7.2.1 A very brief overview of MIDI

MIDI (Musical Instrument Digital Interface) was first introduced in 1981 by Dave Smith, the founder of Sequential Circuits. The original purpose of MIDI was to allow the communication between different electronic musical instruments; more specifically, to allow one device to send **control** data to another device. Typical messages might be “note On” (play a *note*) “note Off” (turn off a *note*). A MIDI “note” message, for example, is composed of three bytes: the first byte (the Status byte) tells the device what kind of message this is (e.g. a Note On message). The second byte encodes key number (which key was pressed) and the third byte, velocity (how hard the key was pressed). It should be clear that a *Note* in this context means something very different than *Note* in the context of a traditional printed score. While the bias towards keyboard interfaces is clear in the definition of the MIDI Note control message, one can still give the MIDI note a more general use by reinterpreting “key number” as pitch and “velocity” as loudness, the usual perceptual correlates of these control changes as well as the most meaningful musical parameters in western music.

With the subsequent proliferation of music production software, the SMF (Standard Midi File) was introduced to allow the recording and storage of the control data from a MIDI stream. The SMF required a time stamp to keep track of when control messages took place. These are called “delta-times” in the SMF specification.

“The MTrk chunk type is where actual song data is stored. It is simply a stream of MIDI events (and non-MIDI events), preceded by delta-time values.”

In combination with the MIDI Note message, the addition of duration now allowed one to have a minimal but sufficient **machine** representation—a machine score—of music requiring only these parameters: duration, pitch and loudness. Such is the case of most piano music.

7.2.2 Limitations of MIDI from the point of view of score modeling

But, alas, there is much more information in a printed score that can not be practically encoded in a SMF. Common musical notions such as meter, clef, key signature, articulation, to name only a few, are ignored. A desire to include some of these concepts in MIDI is evident in the inclusion of some so called *meta-events*. From the SMF specification:

” specifies non-MIDI information useful to this format or to sequencers.” Examples of *meta-events* are *Time Signature* and *Key Signature*. In addition to the semantic elements just mentioned, there are also the typographical elements (such as line thickness, spacing, color, fonts, etc.) that all printed scores carry. This extra layer of information is completely absent in a SMF. However, from the point of view of encoding a printed score, the main limitation of MIDI is not the lack musical features or the absence of typographical data, but the assumption that musical durations, pitches and loudnesses can be each fully and efficiently encoded with integers or even fractions. In a printed score, this is not the case for any of them. MIDI encodes only *magnitudes*: time interval magnitudes, pitch interval magnitudes, velocity magnitudes. While these may be sufficient attributes for an automated piano performance, they are not all the attributes of notes in a printed score.

7.2.3 Written note durations vs. MIDI delta-times

Assume a fixed tempo has been set. Assume that all magnitudes are represented with (and limited to) rational numbers. A time interval magnitude $d = 1/4$ has an infinity of equivalent representations in terms of magnitude: $d = 1/4 = 1/8 * 2 = 1/8 + 1/16 * 2 \dots$ etc. So, for example, while equivalent in magnitude, these are not the same notated durations:

```
>>> m1 = measuretools.AnonymousMeasure([Note("c'4")])
>>> m2 = measuretools.AnonymousMeasure(Note(0, (1, 8)) * 2)
>>> tietools.TieSpanner(m2)
TieSpanner(|1/4(2)|)
>>> m3 = measuretools.AnonymousMeasure([Note(0, (1, 8))] + Note(0, (1, 16)) * 2)
>>> tietools.TieSpanner(m3)
TieSpanner(|1/4(3)|)
>>> r = stafftools.RhythmicStaff([m1, m2, m3])

>>> show(r)
```



7.2.4 Written note pitch vs. MIDI note-on

A similar thing happens with pitches. In MIDI, key (pitch) number 61 is a half tone above middle C. But how is this pitch to be notated? As a C sharp or a B flat?

```
>>> m1 = measuretools.AnonymousMeasure([Note(1, (1, 4))])
>>> m2 = measuretools.AnonymousMeasure([Note(('df', 4), (1, 4))])
>>> r = Staff([m1, m2])

>>> show(r)
```



7.2.5 Conclusion

MIDI was not designed for score representation. MIDI is a simple communication protocol intended for real-time control. As such, it naturally lacks the adequate model to represent the full range of information found in printed scores.

7.3 Why LilyPond is right for Abjad

Early versions of Abjad wrote MIDI files for input to Finale and Sibelius. Later versions of Abjad wrote .pbx files for input into Leland Smith's SCORE. Over time we found LilyPond superior to Finale, Sibelius and SCORE.

7.3.1 Nested tuplets works out of the box

LilyPond uses a single construct to nest tuplets arbitrarily:

```
\new stafftools.RhythmicStaff {
  \time 7/8
  \times 7/8 {
    c8.
    \times 7/5 { c16 c16 c16 c16 c16 }
    \times 3/5 { c8 c8 c8 c8 c8 }
  }
}

>>> staff = stafftools.RhythmicStaff([Measure((7, 8), [])])
>>> measure = staff[0]
>>> measure.append(Note('c8.'))
>>> measure.append(Tuplet(Fraction(7, 5), 5 * Note('c16')))
>>> beamtools.BeamSpanner(measure[-1])
BeamSpanner({c16, c16, c16, c16, c16})
>>> measure.append(Tuplet(Fraction(3, 5), 5 * Note('c8')))
>>> beamtools.BeamSpanner(measure[-1])
BeamSpanner({c8, c8, c8, c8, c8})
>>> Tuplet(Fraction(7, 8), measure.music)
Tuplet(7/8, [c8., {* 5:7 c16, c16, c16, c16, c16 *}, {* 5:3 c8, c8, c8, c8, c8 *}])
>>> staff.override.tuplet_bracket.bracket_visibility = True
>>> staff.override.tuplet_bracket.padding = 1.6

>>> show(staff, docs=True)
```



LilyPond's tuplet input syntax works the same as any other recursive construct.

7.3.2 Broken tuplets work out of the box

LilyPond engraves tupletted notes interrupted by nontupletted notes correctly:

```
\new Staff {
  \times 4/7 { c'16 c'16 c'16 c'16 }
  c'8 c'8
  \times 4/7 { c'16 c'16 c'16 }
}

>>> t = Tuplet(Fraction(4, 7), Note(0, (1, 16)) * 4)
>>> notes = Note(0, (1, 8)) * 2
>>> u = Tuplet(Fraction(4, 7), Note(0, (1, 16)) * 3)
>>> beamtools.BeamSpanner(t)
BeamSpanner({c'16, c'16, c'16, c'16})
```

```
>>> beamtools.BeamSpanner(notes)
BeamSpanner(c'8, c'8)
>>> beamtools.BeamSpanner(u)
BeamSpanner({c'16, c'16, c'16})
>>> measure = Measure((4, 8), [t] + notes + [u])
>>> staff = stafftools.RhythmicStaff([measure])

>>> show(staff, docs=True)
```



7.3.3 Nonbinary meters work out of the box

The rhythm above rewrites with time signatures in place of tuplets:

```
\new Staff {
  \time 4/28 c'16 c'16 c'16 c'16 |
  \time 2/8 c'8 c'8 |
  \time 3/28 c'16 c'16 c'16 |
}

>>> t = Measure((4, 28), Note(0, (1, 16)) * 4)
>>> u = Measure((2, 8), Note(0, (1, 8)) * 2)
>>> v = Measure((3, 28), Note(0, (1, 16)) * 3)
>>> beamtools.BeamSpanner(t)
BeamSpanner(|4/28(4)|)
>>> beamtools.BeamSpanner(u)
BeamSpanner(|2/8(2)|)
>>> beamtools.BeamSpanner(v)
BeamSpanner(|3/28(3)|)
>>> staff = stafftools.RhythmicStaff([t, u, v])

>>> show(staff)
```



The time signatures 4/28 and 3/28 here have a denominator not equal to 4, 8, 16 or any other nonnegative integer power of two. Abjad calls such time signatures **nonbinary meters** and LilyPond engraves them correctly.

7.3.4 Lilypond models the musical measure correctly

Most engraving packages make the concept of the measure out to be more important than it should. We see evidence of this wherever an engraving package makes it difficult for either a long note or the notes of a tuplet to cross a barline. These difficulties come from working the idea of measure-as-container deep into object model of the package.

There is a competing way to model the musical measure that we might call the measure-as-background way of thinking about things. Western notation practice started absent any concept of the barline, introduced the idea gradually, and has since retreated from the necessity of the convention. Engraving packages that pick out an understanding of the barline from the 18th or 19th centuries subscribe to the measure-as-container view of things and oversimplify the problem. One result of this is to render certain barline-crossing rhythmic figures either an inelegant hack or an outright impossibility. LilyPond eschews the measure-as-container model in favor of the measure-as-background model better able to handle both earlier and later notation practice.

7.4 LilyPond text alignment

LilyPond provides many ways to position text.

7.4.1 Default alignment

LilyPond left-aligns markup relative to the left edge of note heads by default.

```
>>> from abjad.tools import documentationtools

>>> staff = stafftools.RhythmicStaff('c')

>>> markuptools.Markup('XX', Up)(staff[0])

>>> lilypond_file = documentationtools.make_text_alignment_example_lilypond_file(staff)
>>> show(lilypond_file)
```



7.4.2 TextScript #'self-alignment-X

Use #'self-alignment-X to left-, center- or right-align markup relative to the left edge of note heads.

Note that changes to #'self-alignment-X do not change the fact that markup positioning is by default relative to the left edge of note heads.

```
>>> staff = stafftools.RhythmicStaff('c c c')

>>> markuptools.Markup('XX', Up)(staff[0])
>>> staff[0].override.text_script.self_alignment_X = 'left'
>>> markuptools.Markup('XX', Up)(staff[1])
>>> staff[1].override.text_script.self_alignment_X = 'center'
>>> markuptools.Markup('XX', Up)(staff[2])
>>> staff[2].override.text_script.self_alignment_X = 'right'

>>> lilypond_file = documentationtools.make_text_alignment_example_lilypond_file(staff)
>>> show(lilypond_file)
```



7.4.3 TextScript #'X-offset

Use #'X-offset to offset markup by some number of magic units in the horizontal direction.

Specify #'X-offset arguments as numbers like #2.5. Do not specify #'X-offset arguments as direction constants like #right.

Note that changes to #'X-offset do not change the fact that markup positioning is by default relative to the left edge of note heads.

```
>>> staff = stafftools.RhythmicStaff('c c c c')
```

```
>>> markuptools.Markup('XX', Up)(staff[0])
>>> staff[0].override.text_script.X_offset = 0
>>> markuptools.Markup('XX', Up)(staff[1])
>>> staff[1].override.text_script.X_offset = 2
>>> markuptools.Markup('XX', Up)(staff[2])
>>> staff[2].override.text_script.X_offset = 4
>>> markuptools.Markup('XX', Up)(staff[3])
>>> staff[3].override.text_script.X_offset = 6

>>> lilypond_file = documentationtools.make_text_alignment_example_lilypond_file(staff)
>>> show(staff)
```



7.5 Score Snippet Gallery

Abjad uses a collection of score snippets in many places throughout the docs, tests and other parts of the codebase. Some of these are collected here.

7.5.1 Score snippet 1

This score features two measures with a beam spanner applied to each measure and a slur spanner applied to all the notes in the score:

```
>>> staff = Staff(r"abj: | 2/8 c'8 d'8 || 2/8 e'8 f'8 |")
>>> beamtools.apply_beam_spanners_to_measures_in_expr(staff)
[BeamSpanner(|2/8(2)|), BeamSpanner(|2/8(2)|)]
>>> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 [ (
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}
```

>>> show(staff)



Score snippet 1 is used widely in the component split tests.

7.6 Change log

7.6.1 Changes from 2.9 to 2.10

Renamed the read-only `format` property to `lilypond_format` on all system objects.

All iteration functions are now housed in the new `iterationtools` package:

- Renamed:

```
chordtools.iterate_chords_forward_in_expr()
chordtools.iterate_chords_backward_in_expr()

iterationtools.iterate_chords_in_expr(reverse=[True, False])
```

- Renamed:

```
componenttools.iterate_components_depth_first()
componenttools.iterate_components_forward_in_expr()
componenttools.iterate_components_backward_in_expr()
componenttools.iterate_namesakes_forward_from_component()
componenttools.iterate_namesakes_backward_from_component()
componenttools.iterate_thread_forward_from_component()
componenttools.iterate_thread_backward_from_component()
componenttools.iterate_thread_forward_in_expr()
componenttools.iterate_thread_backward_in_expr()
componenttools.iterate_timeline_forward_from_component()
componenttools.iterate_timeline_backward_from_component()
componenttools.iterate_timeline_forward_in_expr()
componenttools.iterate_timeline_backward_in_expr()

iterationtools.iterate_components_depth_first()
iterationtools.iterate_components_in_expr(reverse=[True, False])
iterationtools.iterate_namesakes_from_component(reverse=[True, False])
iterationtools.iterate_thread_from_component(reverse=[True, False])
iterationtools.iterate_thread_in_expr(reverse=[True, False])
iterationtools.iterate_timeline_from_component(reverse=[True, False])
iterationtools.iterate_timeline_in_expr(reverse=[True, False])
```

- Renamed:

```
containertools.iterate_containers_forward_in_expr()
containertools.iterate_containers_backward_in_expr()

iterationtools.iterate_containers_in_expr(reverse=[True, False])
```

- Renamed:

```
contexttools.iterate_contexts_forward_in_expr()
contexttools.iterate_contexts_backward_in_expr()

iterationtools.iterate_contexts_in_expr(reverse=[True, False])
```

- Renamed:

```
gracetools.iterate_components_and_grace_containers_forward_in_expr()

iterationtools.iterate_components_and_grace_containers_in_expr()
```


- Renamed:

```
leaftools.iterate_leaf_pairs_forward_in_expr()
leaftools.iterate_leaves_forward_in_expr()
leaftools.iterate_leaves_backward_in_expr()
leaftools.iterate_notes_and_chords_forward_in_expr()
leaftools.iterate_notes_and_chords_backward_in_expr()

iterationtools.iterate_leaf_pairs_in_expr()
iterationtools.iterate_leaves_in_expr(reverse=[True, False])
iterationtools.iterate_notes_and_chords_in_expr(reverse=[True, False])
```

- Renamed:

```
measuretools.iterate_measures_forward_in_expr()
measuretools.iterate_measures_backward_in_expr()

iterationtools.iterate_measures_in_expr(reverse=[True, False])
```

- Renamed:

```
notetools.iterate_notes_forward_in_expr()
notetools.iterate_notes_backward_in_expr()

iterationtools.iterate_notes_in_expr(reverse=[True, False])
```

- Renamed:

```
resttools.iterate_rests_forward_in_expr()
resttools.iterate_rests_backward_in_expr()

iterationtools.iterate_rests_in_expr(reverse=[True, False])
```

- Renamed:

```
scoretools.iterate_scores_forward_in_expr()
scoretools.iterate_scores_backward_in_expr()

iterationtools.iterate_scores_in_expr(reverse=[True, False])
```

- Renamed:

```
skiptools.iterate_skips_forward_in_expr()
skiptools.iterate_skips_backward_in_expr()

iterationtools.iterate_skips_in_expr(reverse=[True, False])
```

- Renamed:

```
stafftools.iterate_staves_forward_in_expr()
stafftools.iterate_staves_backward_in_expr()

iterationtools.iterate_staves_in_expr(reverse=[True, False])
```

- Renamed:

```
tuplettools.iterate_tuplets_forward_in_expr()
tuplettools.iterate_tuplets_backward_in_expr()
```

```
iterationtools.iterate_tuplets_in_expr(reverse=[True, False])
```

- Renamed:

```
voicetools.iterate_semantic_voices_forward_in_expr()
voicetools.iterate_semantic_voices_backward_in_expr()
voicetools.iterate_voices_forward_in_expr()
voicetools.iterate_voices_backward_in_expr()

voicetools.iterate_semantic_voices_in_expr(reverse=[True, False])
voicetools.iterate_voices_in_expr(reverse=[True, False])
```

All labeling functions are now housed in the new `labeltools` package:

- Renamed:

```
chordtools.color_chord_note_heads_in_expr_by_pitch_class_color_map()

labeltools.color_chord_note_heads_in_expr_by_pitch_class_color_map()
```

- Renamed:

```
containertools.color_contents_of_container()

labeltools.color_contents_of_container()
```

- Renamed:

```
leaftools.color_leaf()
leaftools.color_leaves_in_expr()
leaftools.label_leaves_in_expr_with_inversion_equivalent_chromatic_interval_classes()
leaftools.label_leaves_in_expr_with_leaf_depth()
leaftools.label_leaves_in_expr_with_leaf_durations()
leaftools.label_leaves_in_expr_with_leaf_indices()
leaftools.label_leaves_in_expr_with_leaf_numbers()
leaftools.label_leaves_in_expr_with_melodic_chromatic_interval_classes()
leaftools.label_leaves_in_expr_with_melodic_chromatic_intervals()
leaftools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes()
leaftools.label_leaves_in_expr_with_melodic_counterpoint_intervals()
leaftools.label_leaves_in_expr_with_melodic_diatonic_interval_classes()
leaftools.label_leaves_in_expr_with_melodic_diatonic_intervals()
leaftools.label_leaves_in_expr_with_pitch_class_numbers()
leaftools.label_leaves_in_expr_with_pitch_numbers()
leaftools.label_leaves_in_expr_with_prolated_leaf_duration()
leaftools.label_leaves_in_expr_with_tuplet_depth()
leaftools.label_leaves_in_expr_with_written_leaf_duration()

labeltools.color_leaf()
labeltools.color_leaves_in_expr()
labeltools.label_leaves_in_expr_with_inversion_equivalent_chromatic_interval_classes()
labeltools.label_leaves_in_expr_with_leaf_depth()
labeltools.label_leaves_in_expr_with_leaf_durations()
labeltools.label_leaves_in_expr_with_leaf_indices()
labeltools.label_leaves_in_expr_with_leaf_numbers()
labeltools.label_leaves_in_expr_with_melodic_chromatic_interval_classes()
labeltools.label_leaves_in_expr_with_melodic_chromatic_intervals()
labeltools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes()
labeltools.label_leaves_in_expr_with_melodic_counterpoint_intervals()
labeltools.label_leaves_in_expr_with_melodic_diatonic_interval_classes()
labeltools.label_leaves_in_expr_with_melodic_diatonic_intervals()
```

```
labeltools.label_leaves_in_expr_with_pitch_class_numbers()
labeltools.label_leaves_in_expr_with_pitch_numbers()
labeltools.label_leaves_in_expr_with_prolated_leaf_duration()
labeltools.label_leaves_in_expr_with_tuplet_depth()
labeltools.label_leaves_in_expr_with_written_leaf_duration()
```

- Renamed:

```
markuptools.remove_markup_from_leaves_in_expr()

labeltools.remove_markup_from_leaves_in_expr()
```

- Renamed:

```
measuretools.color_measure()
measuretools.color_nonbinary_measures_in_expr()

labeltools.color_measure()
labeltools.color_nonbinary_measures_in_expr()
```

- Renamed:

```
notetools.color_note_head_by_numbered_chromatic_pitch_class_color_map()
notetools.label_notes_in_expr_with_note_indices()

labeltools.color_note_head_by_numbered_chromatic_pitch_class_color_map()
labeltools.label_notes_in_expr_with_note_indices()
```

- Renamed:

```
tietools.label_tie_chains_in_expr_with_prolated_tie_chain_duration()
tietools.label_tie_chains_in_expr_with_tie_chain_durations()
tietools.label_tie_chains_in_expr_with_written_tie_chain_duration()

labeltools.label_tie_chains_in_expr_with_prolated_tie_chain_duration()
labeltools.label_tie_chains_in_expr_with_tie_chain_durations()
labeltools.label_tie_chains_in_expr_with_written_tie_chain_duration()
```

- Renamed:

```
verticalitytools.label_vertical_moments_in_expr_with_chromatic_interval_classes()
verticalitytools.label_vertical_moments_in_expr_with_chromatic_intervals()
verticalitytools.label_vertical_moments_in_expr_with_counterpoint_intervals()
verticalitytools.label_vertical_moments_in_expr_with_diatonic_intervals()
verticalitytools.label_vertical_moments_in_expr_with_interval_class_vectors()
verticalitytools.label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes()
verticalitytools.label_vertical_moments_in_expr_with_pitch_numbers()

labeltools.label_vertical_moments_in_expr_with_chromatic_interval_classes()
labeltools.label_vertical_moments_in_expr_with_chromatic_intervals()
labeltools.label_vertical_moments_in_expr_with_counterpoint_intervals()
labeltools.label_vertical_moments_in_expr_with_diatonic_intervals()
labeltools.label_vertical_moments_in_expr_with_interval_class_vectors()
labeltools.label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes()
labeltools.label_vertical_moments_in_expr_with_pitch_numbers()
```

Renamed all functions that contained `big_endian`:

```
durationtools.duration_token_to_big_endian_list_of_assignable_duration_pairs()
leaftools.fuse_leaves_big_endian()
leaftools.fuse_leaves_in_tie_chain_by_immediate_parent_big_endian()
```

```
durationtools.duration_token_to_assignable_duration_pairs()
leaftools.fuse_leaves()
leaftools.fuse_leaves_in_tie_chain_by_immediate_parent()
```

Renamed all functions that contained `prolated_offset` to simply `offset`:

```
componenttools.copy_governed_component_subtree_from_prolated_offset_to()
componenttools.get_improper_descendents_of_component_that_cross_prolated_offset()
containertools.delete_contents_of_container_starting_at_or_after_prolated_offset()
containertools.delete_contents_of_container_starting_before_or_at_prolated_offset()
containertools.delete_contents_of_container_starting_strictly_after_prolated_offset()
containertools.delete_contents_of_container_starting_strictly_before_prolated_offset()
containertools.get_element_starting_at_exactly_prolated_offset()
containertools.get_first_element_starting_at_or_after_prolated_offset()
containertools.get_first_element_starting_before_or_at_prolated_offset()
containertools.get_first_element_starting_strictly_after_prolated_offset()
containertools.get_first_element_starting_strictly_before_prolated_offset()
prolated_offsettools.update_offset_values_of_component()
verticalitytools.get_vertical_moment_at_prolated_offset_in_expr()
```

```
componenttools.copy_governed_component_subtree_from_offset_to()
componenttools.get_improper_descendents_of_component_that_cross_offset()
containertools.delete_contents_of_container_starting_at_or_after_offset()
containertools.delete_contents_of_container_starting_before_or_at_offset()
containertools.delete_contents_of_container_starting_strictly_after_offset()
containertools.delete_contents_of_container_starting_strictly_before_offset()
containertools.get_element_starting_at_exactly_offset()
containertools.get_first_element_starting_at_or_after_offset()
containertools.get_first_element_starting_before_or_at_offset()
containertools.get_first_element_starting_strictly_after_offset()
containertools.get_first_element_starting_strictly_before_offset()
offsettools.update_offset_values_of_component()
verticalitytools.get_vertical_moment_at_offset_in_expr()
```

Renamed `prolated_duration` to `offset` in some functions:

```
componenttools.split_component_at_prolated_duration()
componenttools.split_components_by_prolated_durations()
leaftools.split_leaf_at_prolated_duration()
leaftools.split_leaf_at_prolated_duration_and_rest_right_half()
```

```
componenttools.split_component_at_offset()
componenttools.split_components_by_offsets()
leaftools.split_leaf_at_offset()
leaftools.split_leaf_at_offset_and_rest_right_half()
```

Renamed all functions that contained `as_string`:

```
componenttools.report_component_format_contributions_as_string()
containertools.report_container_modifications_as_string()
measuretools.report_meter_distribution_as_string()
```

```
componenttools.report_component_format_contributions()
containertools.report_container_modifications()
measuretools.report_time_signature_distribution()
```

Changes to the `componenttools` package:

- The `componenttools.split_components_at_offsets()` function no longer implements a `tie_after` keyword. Use the new `tie_split_notes` and `tie_split_rests` keywords. Note that the new `tie_split_rests` keyword defaults to `true` where the old `tie_after` keyword defaulted to `false`. This changes the default behavior of the function.

- Renamed:

```
componenttools.extend_left_in_parent_of_component_and_grow_spanners()
componenttools.extend_left_in_parent_of_component_and_do_not_grow_spanners()

componenttools.extend_left_in_parent_of_component(grow_spanners=[True, False])
```

- Renamed:

```
componenttools.extend_in_parent_of_component_and_grow_spanners()
componenttools.extend_in_parent_of_component_and_do_not_grow_spanners()

componenttools.extend_in_parent_of_component(grow_spanners=[True, False])
```

- Renamed:

```
componenttools.number_is_between_prolated_start_and_stop_offsets_of_component()

componenttools.number_is_between_start_and_stop_offsets_of_component()
```

- Renamed:

```
componenttools.partition_components_cyclically_by_durations_in_seconds_exactly_with_overhang()
componenttools.partition_components_cyclically_by_durations_in_seconds_exactly_without_overhang()
componenttools.partition_components_cyclically_by_durations_in_seconds_ge_with_overhang()
componenttools.partition_components_cyclically_by_durations_in_seconds_ge_without_overhang()
componenttools.partition_components_cyclically_by_durations_in_seconds_le_with_overhang()
componenttools.partition_components_cyclically_by_durations_in_seconds_le_without_overhang()
componenttools.partition_components_cyclically_by_prolated_durations_exactly_with_overhang()
componenttools.partition_components_cyclically_by_prolated_durations_exactly_without_overhang()
componenttools.partition_components_cyclically_by_prolated_durations_ge_with_overhang()
componenttools.partition_components_cyclically_by_prolated_durations_ge_without_overhang()
componenttools.partition_components_cyclically_by_prolated_durations_le_with_overhang()
componenttools.partition_components_cyclically_by_prolated_durations_le_without_overhang()
componenttools.partition_components_once_by_durations_in_seconds_exactly_with_overhang()
componenttools.partition_components_once_by_durations_in_seconds_exactly_without_overhang()
componenttools.partition_components_once_by_durations_in_seconds_ge_with_overhang()
componenttools.partition_components_once_by_durations_in_seconds_ge_without_overhang()
componenttools.partition_components_once_by_durations_in_seconds_le_with_overhang()
componenttools.partition_components_once_by_durations_in_seconds_le_without_overhang()
componenttools.partition_components_once_by_prolated_durations_exactly_with_overhang()
componenttools.partition_components_once_by_prolated_durations_exactly_without_overhang()
componenttools.partition_components_once_by_prolated_durations_ge_with_overhang()
componenttools.partition_components_once_by_prolated_durations_ge_without_overhang()
componenttools.partition_components_once_by_prolated_durations_le_with_overhang()
componenttools.partition_components_once_by_prolated_durations_le_without_overhang()

componenttools.partition_components_by_durations_exactly()
componenttools.partition_components_by_durations_ge()
componenttools.partition_components_by_durations_le()
```

- Renamed:

```
componenttools.split_component_at_prolated_duration_and_do_not_fracture_crossing_spanners()  
componenttools.split_component_at_prolated_duration_and_fracture_crossing_spanners()
```

```
componenttools.split_component_at_offset(fracture_spanners=[True, False])
```

- **Renamed:**

```
componenttools.split_components_cyclically_by_prolated_durations_and_do_not_fracture_crossing_sp  
componenttools.split_components_cyclically_by_prolated_durations_and_fracture_crossing_spanners()  
componenttools.split_components_once_by_prolated_durations_and_do_not_fracture_crossing_spanners()  
componenttools.split_components_once_by_prolated_durations_and_fracture_crossing_spanners()
```

```
componenttools.split_components_at_offsets(fracture_spanners=[True, False], cyclic=[True, False])
```

Changeds to the `containertools` package:

- **Renamed:**

```
containertools.remove_empty_containers_in_expr()
```

```
containertools.remove_leafless_containers_in_expr()
```

- **Renamed:**

```
containertools.replace_larger_left_half_of_elements_in_container_with_big_endian_rests()  
containertools.replace_larger_left_half_of_elements_in_container_with_little_endian_rests()  
containertools.replace_larger_right_half_of_elements_in_container_with_big_endian_rests()  
containertools.replace_larger_right_half_of_elements_in_container_with_little_endian_rests()  
containertools.replace_n_edge_elements_in_container_with_big_endian_rests()  
containertools.replace_n_edge_elements_in_container_with_little_endian_rests()  
containertools.replace_n_edge_elements_in_container_with_rests()  
containertools.replace_smaller_left_half_of_elements_in_container_with_big_endian_rests()  
containertools.replace_smaller_left_half_of_elements_in_container_with_little_endian_rests()  
containertools.replace_smaller_right_half_of_elements_in_container_with_big_endian_rests()  
containertools.replace_smaller_right_half_of_elements_in_container_with_little_endian_rests()
```

```
containertools.replace_container_slice_with_rests()
```

- **Renamed:**

```
containertools.split_container_at_index_and_do_not_fracture_crossing_spanners()  
containertools.split_container_at_index_and_fracture_crossing_spanners()
```

```
containertools.split_container_at_index(fracture_spanners=[True, False])
```

- **Renamed:**

```
containertools.split_container_cyclically_by_counts_and_do_not_fracture_crossing_spanners()  
containertools.split_container_cyclically_by_counts_and_fracture_crossing_spanners()  
containertools.split_container_once_by_counts_and_do_not_fracture_crossing_spanners()  
containertools.split_container_once_by_counts_and_fracture_crossing_spanners()
```

```
containertools.split_container_by_counts(fracture_spanners=[True, False], cyclic=[True, False])
```

Changes to the `durationtools` package:

- **Renamed:**

```
durationtools.yield_all_assignable_rationals_in_cantor_diagonalized_order()  
durationtools.yield_all_positive_integer_pairs_in_cantor_diagonalized_order()
```

```

durationtools.yield_all_positive_rationals_in_cantor_diagonalized_order()
durationtools.yield_all_positive_rationals_in_cantor_diagonalized_order_uniquely()
durationtools.yield_all_prolation_rewrite_pairs_of_rational_in_cantor_diagonalized_order()

durationtools.yield_all_assignable_rationals()
durationtools.yield_all_positive_integer_pairs()
durationtools.yield_all_positive_rationals()
durationtools.yield_all_positive_rationals_uniquely()
durationtools.yield_prolation_rewrite_pairs()

```

Changes to the instrumenttools package:

- Renamed:

```

instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch()

instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch()

```

- Renamed:

```

instrumenttools.transpose_notes_and_chords_in_expr_from_fingered_pitch_to_sounding_pitch()

instrumenttools.transpose_from_fingered_pitch_to_sounding_pitch()

```

Changes to the leaftools package:

- Renamed:

```

leaftools.fuse_leaves_in_container_once_by_counts_into_big_endian_notes()
leaftools.fuse_leaves_in_container_once_by_counts_into_big_endian_rests()
leaftools.fuse_leaves_in_container_once_by_counts_into_little_endian_notes()
leaftools.fuse_leaves_in_container_once_by_counts_into_little_endian_rests()

leaftools.fuse_leaves_in_container_once_by_counts(big_endian=[True, False], klass=None)

```

- Renamed:

```

leaftools.leaf_to_augmented_tuplet_with_n_notes_of_equal_written_duration()
leaftools.leaf_to_augmented_tuplet_with_proportions()
leaftools.leaf_to_diminished_tuplet_with_n_notes_of_equal_written_duration()
leaftools.leaf_to_diminished_tuplet_with_proportions()

tuplettools.leaf_to_tuplet_with_n_notes_of_equal_written_duration()
tuplettools.leaf_to_tuplet_with_proportions()

```

- Renamed:

```

leaftools.split_leaf_at_offset_and_rest_right_half()

leaftools.rest_leaf_at_offset()

```

- Renamed:

```

leaftools.repeat_leaf_and_extend_spanners()
leaftools.repeat_leaves_in_expr_and_extend_spanners()

leaftools.repeat_leaf()
leaftools.repeat_leaves_in_expr()

```

Changes to the mathtools package.

- Removed `mathtools.partition_integer_into_thirds()`.

Changes to the `measuretools` package:

- Renamed:

```
measuretools.fill_measures_in_expr_with_meter_denominator_notes()
measuretools.move_prolation_of_full_measure_tuplet_to_meter_of_measure()
measuretools.multiply_contents_of_measures_in_expr_and_scale_meter_denominators()
measuretools.scale_measure_by_multiplier_and_adjust_meter()

measuretools.fill_measures_in_expr_with_time_signature_denominator_notes()
measuretools.move_full_measure_tuplet_prolation_to_measure_time_signature()
measuretools.multiply_contents_of_measures_in_expr_and_scale_time_signature_denominators()
measuretools.scale_measure_and_adjust_time_signature()
```

- Renamed:

```
measuretools.fill_measures_in_expr_with_big_endian_notes()
measuretools.fill_measures_in_expr_with_little_endian_notes()

measuretools.measuretools.fill_measures_in_expr_with_minimal_number_of_notes(big_endian=[True, False])
```

- Renamed:

```
measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets_to_measure_contents()

measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets()
```

- Renamed:

```
measuretools.get_prev_measure_from_component()

measuretools.get_previous_measure_from_component()
```

- Renamed:

```
measuretools.multiply_contents_of_measures_in_expr_and_scale_time_signature_denominators()

measuretools.multiply_and_scale_contents_of_measures_in_expr()
```

- Renamed:

```
measuretools.pitch_array_row_to_measure()
measuretools.pitch_array_to_measures()

pitchtools.pitch_array_row_to_measure()
pitchtools.pitch_array_to_measures()
```

Changes to the `pitchtools` package:

- Renamed:

```
pitchtools.calculate_harmonic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier()
pitchtools.calculate_harmonic_chromatic_interval_from_pitch_carrier_to_pitch_carrier()
pitchtools.calculate_harmonic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch()
pitchtools.calculate_harmonic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch()
pitchtools.calculate_harmonic_diatonic_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch()
pitchtools.calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch()
```



```
pitchtools.calculate_harmonic_chromatic_interval_class()
pitchtools.calculate_harmonic_chromatic_interval()
pitchtools.calculate_harmonic_counterpoint_interval_class()
pitchtools.calculate_harmonic_counterpoint_interval()
pitchtools.calculate_harmonic_diatonic_interval_class()
pitchtools.calculate_harmonic_diatonic_interval()
```

- **Renamed:**

```
pitchtools.calculate_melodic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier()
pitchtools.calculate_melodic_chromatic_interval_from_pitch_carrier_to_pitch_carrier()
pitchtools.calculate_melodic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch()
pitchtools.calculate_melodic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch()
pitchtools.calculate_melodic_diatonic_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch()
pitchtools.calculate_melodic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch()

pitchtools.calculate_melodic_chromatic_interval_class()
pitchtools.calculate_melodic_chromatic_interval()
pitchtools.calculate_melodic_counterpoint_interval_class()
pitchtools.calculate_melodic_counterpoint_interval()
pitchtools.calculate_melodic_diatonic_interval_class()
pitchtools.calculate_melodic_diatonic_interval()
```

- **Renamed:**

```
pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name_alphabetic_accidental_abbreviation()
pitchtools.split_chromatic_pitch_class_name()
```

- **Renamed:**

```
pitchtools.diatonic_interval_number_and_chromatic_interval_number_to_melodic_diatonic_interval()
pitchtools.spell_chromatic_interval_number()
```

- **Renamed:**

```
pitchtools.named_chromatic_pitches_to_harmonic_chromatic_interval_class_number_dictionary()
pitchtools.harmonic_chromatic_interval_class_number_dictionary()
```

- **Renamed:**

```
pitchtools.chromatic_pitch_number_diatonic_pitch_class_name_to_alphabetic_accidental_abbreviation()
pitchtools.chromatic_pitch_number_diatonic_pitch_class_name_to_accidental_octave_number_pair()
```

- **Renamed:**

```
pitchtools.list_named_chromatic_pitch_carriers_in_expr_sorted_by_numbered_chromatic_pitch_class()
pitchtools.sort_named_chromatic_pitch_carriers_in_expr()
```

- **Renamed:**

```
pitchtools.named_chromatic_pitches_to_inversion_equivalent_chromatic_interval_class_number_dictionary()
pitchtools.inversion_equivalent_chromatic_interval_class_number_dictionary()
```

- **Renamed:**

```
pitchtools.transpose_chromatic_pitch_class_number_by_octaves_to_nearest_neighbor_of_chromatic_pi
```

```
pitchtools.transpose_chromatic_pitch_class_number_to_neighbor_of_chromatic_pitch_number()
```

- **Renamed:**

```
pitchtools.ordered_chromatic_pitch_class_numbers_are_within_ordered_chromatic_pitch_numbers()
```

```
pitchtools.contains_subsegment()
```

- **Renamed:**

```
pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise_between_pitch_carriers()
```

```
pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise()
```

- **Renamed:**

```
pitchtools.list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers()
```

```
pitchtools.list_melodic_chromatic_interval_numbers_pairwise()
```

- **Renamed:**

```
pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_name_accidental_octave_number_triple()
```

```
pitchtools.chromatic_pitch_number_to_chromatic_pitch_triple()
```

- **Renamed:**

```
pitchtools.apply_octavation_spanner_to_pitched_components()
```

```
spannertools.apply_octavation_spanner_to_pitched_components()
```

- **Renamed:**

```
pitchtools.set_ascending_named_chromatic_pitches_on_nontied_pitched_components_in_expr()
```

```
pitchtools.set_ascending_named_chromatic_pitches_on_tie_chains_in_expr()
```

- **Renamed:**

```
pitchtools.set_ascending_diatonic_pitches_on_nontied_pitched_components_in_expr()
```

```
pitchtools.set_ascending_diatonic_pitches_on_tie_chains_in_expr()
```

- **Renamed:**

```
pitchtools.transpose_chromatic_pitch_class_number_to_neighbor_of_chromatic_pitch_number()
```

```
pitchtools.transpose_chromatic_pitch_class_number_chromatic_pitch_number_neighbor()
```

Changes to the `rhythmtreetools` package:

- **Renamed:**

```
rhythmtreetools.parse_reduced_ly_syntax()
```

```
lilypondparsertools.parse_reduced_ly_syntax()
```

Changes to the `scoretemplatetools` package:

- Renamed:

```
scoretemplatetools.GroupedRhythmicStavesScoreTemplate.n
```

```
scoretemplatetools.GroupedRhythmicStavesScoreTemplate.staff_count
```

Changes to the `scoretools` package:

- Renamed:

```
scoretools.make_pitch_array_score_from_pitch_arrays()
```

```
pitchtools.make_pitch_array_score_from_pitch_arrays()
```

Changes to the `sequencetools` package:

- Renamed:

```
sequencetools.partition_sequence_cyclically_by_counts_with_overhang()
```

```
sequencetools.partition_sequence_cyclically_by_counts_without_overhang()
```

```
sequencetools.partition_sequence_once_by_counts_with_overhang()
```

```
sequencetools.partition_sequence_once_by_counts_without_overhang()
```

```
sequencetools.partition_sequence_by_counts(cyclic=[True, False], overhang=[True, False])
```

- Renamed:

```
sequencetools.partition_sequence_extended_to_counts_with_overhang()
```

```
sequencetools.partition_sequence_extended_to_counts_without_overhang()
```

```
sequencetools.partition_sequence_extended_to_counts(overhang=[True, False])
```

- Renamed:

```
sequencetools.partition_sequence_cyclically_by_weights_at_least_with_overhang()
```

```
sequencetools.partition_sequence_cyclically_by_weights_at_least_without_overhang()
```

```
sequencetools.partition_sequence_once_by_weights_at_least_with_overhang()
```

```
sequencetools.partition_sequence_once_by_weights_at_least_without_overhang()
```

```
sequencetools.partition_sequence_by_weights_at_least()
```

- Renamed:

```
sequencetools.partition_sequence_cyclically_by_weights_at_most_with_overhang()
```

```
sequencetools.partition_sequence_cyclically_by_weights_at_most_without_overhang()
```

```
sequencetools.partition_sequence_once_by_weights_at_most_with_overhang()
```

```
sequencetools.partition_sequence_once_by_weights_at_most_without_overhang()
```

```
sequencetools.partition_sequence_by_weights_at_most()
```

- Renamed:

```
sequencetools.partition_sequence_cyclically_by_weights_at_exactly_with_overhang()
```

```
sequencetools.partition_sequence_cyclically_by_weights_at_exactly_without_overhang()
```

```
sequencetools.partition_sequence_once_by_weights_at_exactly_with_overhang()
```

```
sequencetools.partition_sequence_once_by_weights_at_exactly_without_overhang()
```

```
sequencetools.partition_sequence_by_weights_at_exactly()
```

- Renamed:

```
sequencetools.split_sequence_cyclically_by_weights_with_overhang()
sequencetools.split_sequence_cyclically_by_weights_without_overhang()
sequencetools.split_sequence_once_by_weights_with_overhang()
sequencetools.split_sequence_once_by_weights_without_overhang()
```

```
sequencetools.split_sequence_by_weights()
```

- **Renamed:**

```
sequencetools.split_sequence_extended_to_weights_with_overhang()
sequencetools.split_sequence_extended_to_weights_without_overhang()
```

```
sequencetools.split_sequence_extended_to_weights()
```

Changes to the `tietools` package:

- **Renamed:**

```
tietools.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots()
tietools.tie_chain_to_augmented_tuplet_with_proportions_and_encourage_dots()
tietools.tie_chain_to_diminished_tuplet_with_proportions_and_avoid_dots()
tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots()
```

```
tietools.tie_chain_to_tuplet_with_proportions()
```

- **Renamed:**

```
tietools.iterate_nontrivial_tie_chains_forward_in_expr()
tietools.iterate_nontrivial_tie_chains_backward_in_expr()
tietools.iterate_pitched_tie_chains_forward_in_expr()
tietools.iterate_pitched_tie_chains_backward_in_expr()
tietools.iterate_tie_chains_forward_in_expr()
tietools.iterate_tie_chains_backward_in_expr()
```

```
tietools.iterate_nontrivial_tie_chains_in_expr(reverse=[True, False])
tietools.iterate_pitched_tie_chains_in_expr(reverse=[True, False])
tietools.iterate_tie_chains_in_expr(reverse=[True, False])
```

Changes to the `tuplettools` package:

- **Renamed:**

```
tuplettools.is_proper_tuplet_multiplier()
```

```
durationtools.is_proper_tuplet_multiplier()
```

- **Renamed:**

```
tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_avoid_dots()
tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_avoid_dots()
tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_encourage_dots()
tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_encourage_dots()
```

```
tuplettools.make_tuplet_from_durations_and_proportions(big_endian=[True, False])
```

Removed three packages.

- Removed `constrainttools` package.
- Removed `lyricstools` package.
- Removed `quantizationtools` package.

7.7 Bibliography

ABJAD API

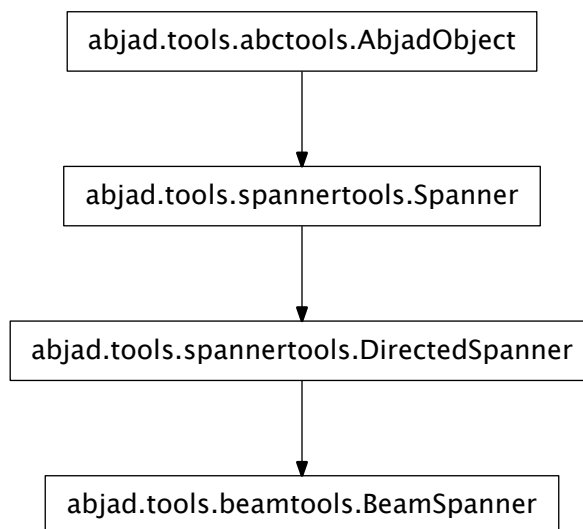
8.1 Abjad API

8.1.1 Composition packages

`beamtools`

concrete classes

`beamtools.BeamSpanner`



```
class beamtools .BeamSpanner (components=None, direction=None)  
    Abjad beam spanner:
```

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'2")

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'2
}

>>> beamtools.BeamSpanner(staff[:4])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
    g'2
}
```

Return beam spanner.

Read-only Properties

BeamSpanner.components

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

BeamSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

BeamSpanner.leaves

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use `spanner-` specific iteration tools.

Inherited from `spannertools.Spanner`

BeamSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

BeamSpanner.preprolated_duration
Sum of preprolated duration of all components in spanner.
Inherited from `spannertools.Spanner`

BeamSpanner.prolated_duration
Sum of prolated duration of all components in spanner.
Inherited from `spannertools.Spanner`

BeamSpanner.set
LilyPond context setting component plug-in.
Inherited from `spannertools.Spanner`

BeamSpanner.start_offset
Read-only start offset of spanner.
Inherited from `spannertools.Spanner`

BeamSpanner.stop_offset
Read-only stop offset of spanner.
Inherited from `spannertools.Spanner`

BeamSpanner.storage_format
Storage format of Abjad object.
Return string.
Inherited from `abctools.AbjadObject`

BeamSpanner.written_duration
Sum of written duration of all components in spanner.
Inherited from `spannertools.Spanner`

Read/write Properties

BeamSpanner.direction
Inherited from `spannertools.DirectedSpanner`

Methods

BeamSpanner.append(*component*)
Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

BeamSpanner.append_left(*component*)
Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`BeamSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`BeamSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`BeamSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`BeamSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`BeamSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`BeamSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`BeamSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`BeamSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`BeamSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`BeamSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`BeamSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`BeamSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BeamSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`BeamSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BeamSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BeamSpanner.__len__()`

Inherited from `spannertools.Spanner`

`BeamSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

`BeamSpanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

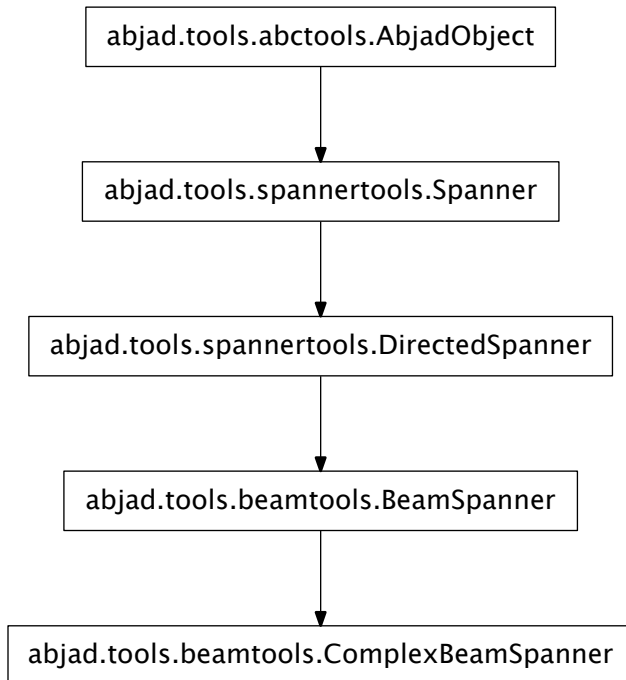
Return boolean.

Inherited from `abctools.AbjadObject`

`BeamSpanner.__repr__()`

Inherited from `spannertools.Spanner`

beamtools.ComplexBeamSpanner



class beamtools.**ComplexBeamSpanner** (*components=None, lone=False, direction=None*)
 Abjad complex beam spanner:

```

>>> staff = Staff("c'16 e'16 r16 f'16 g'2")

>>> f(staff)
\new Staff {
  c'16
  e'16
  r16
  f'16
  g'2
}

>>> beamtools.ComplexBeamSpanner(staff[:4])
ComplexBeamSpanner(c'16, e'16, r16, f'16)

>>> f(staff)
\new Staff {
  \set stemLeftBeamCount = #0
  \set stemRightBeamCount = #2
  c'16 [
  \set stemLeftBeamCount = #2
  \set stemRightBeamCount = #2
  e'16 ]
  
```

```

r16
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #0
f'16 [ ]
g'2
}

```

Return complex beam spanner.

Read-only Properties

`ComplexBeamSpanner.components`

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.leaves`

Return read-only tuple of leaves in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`ComplexBeamSpanner.direction`

Inherited from `spannertools.DirectedSpanner`

`ComplexBeamSpanner.lone`

Beam lone leaf and force beam nibs to left:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone='left')

>>> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #0
c'16 [ ]
```

Beam lone leaf and force beam nibs to right:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone='right')

>>> f(note)
\set stemLeftBeamCount = #0
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and force beam nibs to both left and right:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone='both')

>>> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and accept LilyPond default nibs at both left and right:

```
>>> note = Note("c'16")
```



```
>>> beam = beamtools.ComplexBeamSpanner([note], lone=True)
```

```
>>> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Do not beam lone leaf:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone=False)

>>> f(note)
c'16
```

Set to 'left', 'right', 'both', true or false as shown above.

Ignore this setting when spanner contains more than one leaf.

Methods

`ComplexBeamSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.extend(components)`
Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.extend_left(components)`
Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.fracture(i, direction=None)`
Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.pop()
Note('f'8)
```

```
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.pop_left()
Note("c'8")
```

```
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`ComplexBeamSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.__getitem__(expr)`
 Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

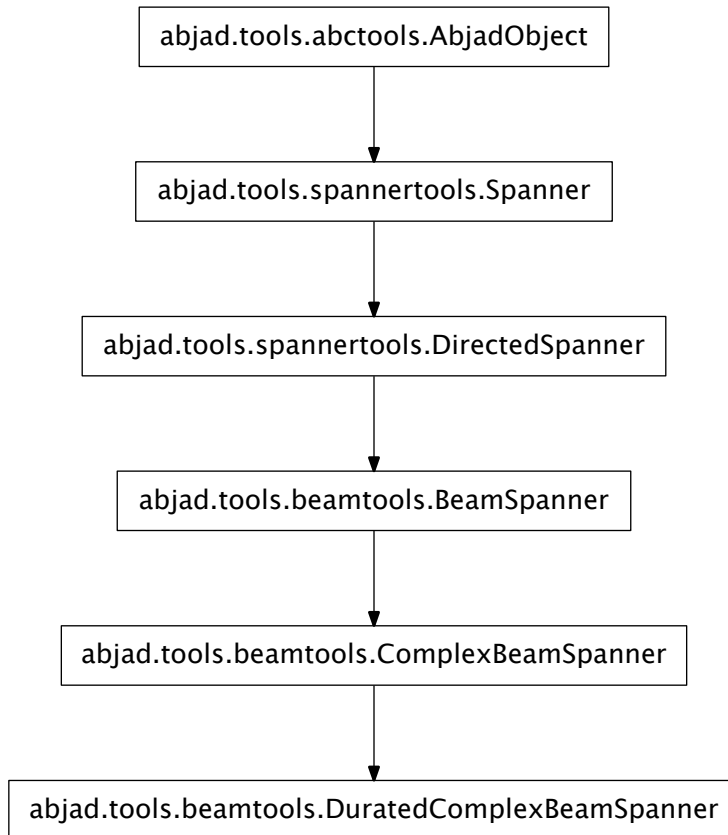
`ComplexBeamSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

beamtools.DuratedComplexBeamSpanner



```

class beamtools.DuratedComplexBeamSpanner(components=None, durations=None, span=1,
                                           lone=False, direction=None)

```

Abjad durated complex beam spanner:

```

staff = Staff("c'16 d'16 e'16 f'16")

durations = [Duration(1, 8), Duration(1, 8)]
beam = beamtools.DuratedComplexBeamSpanner(staff[:], durations, 1)

f(staff)
\new Staff {
  \set stemLeftBeamCount = #0
  \set stemRightBeamCount = #2
  c'16 [
  \set stemLeftBeamCount = #2
  \set stemRightBeamCount = #1
  d'16
  \set stemLeftBeamCount = #1
  \set stemRightBeamCount = #2
  e'16
}

```

```

    \set stemLeftBeamCount = #2
    \set stemRightBeamCount = #0
    f'16 ]
}

```

Beam all beamable leaves in spanner explicitly.

Group leaves in spanner according to *durations*.

Span leaves between duration groups according to *span*.

Return durated complex beam spanner.

Read-only Properties

`DuratedComplexBeamSpanner.components`

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.leaves`

Return read-only tuple of leaves in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`DuratedComplexBeamSpanner.direction`

Inherited from `spannertools.DirectedSpanner`

`DuratedComplexBeamSpanner.durations`

Get spanner leaf group durations:

```
>>> staff = Staff("c'16 d'16 e'16 f'16")
>>> durations = [Duration(1, 8), Duration(1, 8)]
>>> beam = beamtools.DuratedComplexBeamSpanner(staff[:], durations)
>>> beam.durations
[Duration(1, 8), Duration(1, 8)]
```

Set spanner leaf group durations:

```
>>> staff = Staff("c'16 d'16 e'16 f'16")
>>> durations = [Duration(1, 8), Duration(1, 8)]
>>> beam = beamtools.DuratedComplexBeamSpanner(staff[:], durations)
>>> beam.durations = [Duration(1, 4)]
>>> beam.durations
[Duration(1, 4)]
```

Set iterable.

`DuratedComplexBeamSpanner.lone`

Beam lone leaf and force beam nibs to left:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone='left')

>>> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #0
c'16 [ ]
```

Beam lone leaf and force beam nibs to right:

```
>>> note = Note("c'16")
```



```
>>> beam = beamtools.ComplexBeamSpanner([note], lone='right')
```

```
>>> f(note)
\set stemLeftBeamCount = #0
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and force beam nibs to both left and right:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone='both')

>>> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and accept LilyPond default nibs at both left and right:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone=True)

>>> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Do not beam lone leaf:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone=False)

>>> f(note)
c'16
```

Set to 'left', 'right', 'both', true or false as shown above.

Ignore this setting when spanner contains more than one leaf.

Inherited from `beamtools.ComplexBeamSpanner`

`DuratedComplexBeamSpanner`. **span**

Get top-level beam count:

```
>>> staff = Staff("c'16 d'16 e'16 f'16")
>>> durations = [Duration(1, 8), Duration(1, 8)]
>>> beam = beamtools.DuratedComplexBeamSpanner(staff[:], durations, 1)
>>> beam.span
1
```

Set top-level beam count:

```
>>> staff = Staff("c'16 d'16 e'16 f'16")
>>> durations = [Duration(1, 8), Duration(1, 8)]
>>> beam = beamtools.DuratedComplexBeamSpanner(staff[:], durations, 1)
>>> beam.span = 2
>>> beam.span
2
```

Set nonnegative integer.

Methods

`DuratedComplexBeamSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.index(component)`
Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.pop()`
Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.pop_left()`
Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
```

```
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`DuratedComplexBeamSpanner.__call__(expr)`

New in version 2.9. Call `spanner` on *expr* as a shortcut to extend `spanner`:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying `spanner` and mark attachment syntax.

Return `spanner`.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__len__()`

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

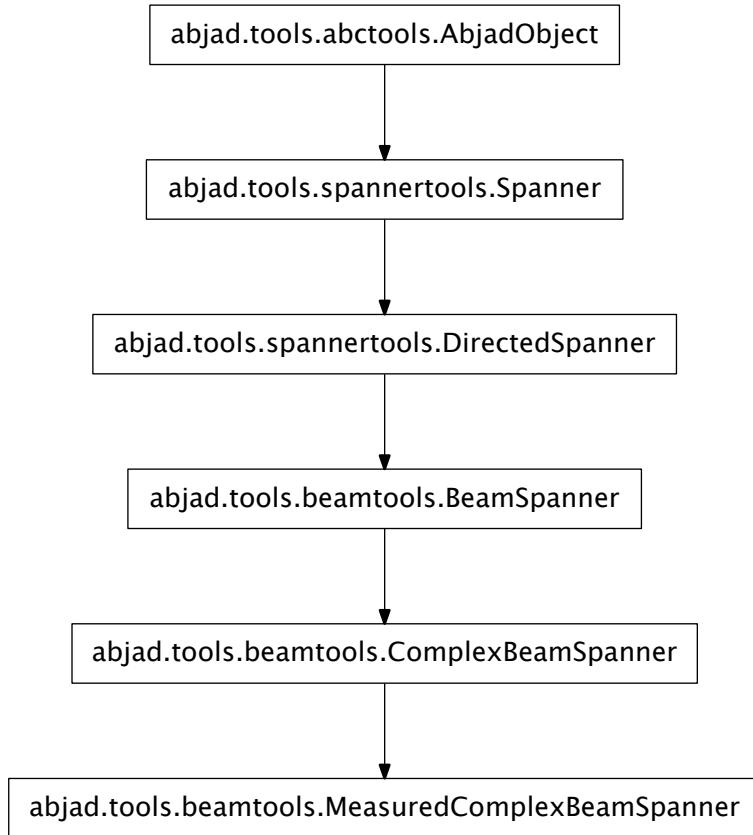
Return boolean.

Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__repr__()`

Inherited from `spannertools.Spanner`

beamtools.MeasuredComplexBeamSpanner



class beamtools.**MeasuredComplexBeamSpanner** (*components=None, lone=False, span=1, direction=None*)

Abjad measured complex beam spanner:

```
>>> staff = Staff([Measure((2, 16), "c'16 d'16"), Measure((2, 16), "e'16 f'16")])
```

```
>>> beamtools.MeasuredComplexBeamSpanner(staff.leaves)
MeasuredComplexBeamSpanner(c'16, d'16, e'16, f'16)
```

```
>>> f(staff)
\new Staff {
  {
    \time 2/16
    \set stemLeftBeamCount = #0
    \set stemRightBeamCount = #2
    c'16 [
    \set stemLeftBeamCount = #2
    \set stemRightBeamCount = #1
    d'16
  }
}
```

```
{
    \set stemLeftBeamCount = #1
    \set stemRightBeamCount = #2
    e'16
    \set stemLeftBeamCount = #2
    \set stemRightBeamCount = #0
    f'16 ]
}
```

Beam leaves in spanner explicitly.

Group leaves by measures.

Format top-level *span* beam between measures.

Return measured complex beam spanner.

Read-only Properties

`MeasuredComplexBeamSpanner.components`

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner-specific iteration tools.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`MeasuredComplexBeamSpanner.direction`

Inherited from `spannertools.DirectedSpanner`

`MeasuredComplexBeamSpanner.lone`

Beam lone leaf and force beam nibs to left:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone='left')

>>> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #0
c'16 [ ]
```

Beam lone leaf and force beam nibs to right:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone='right')

>>> f(note)
\set stemLeftBeamCount = #0
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and force beam nibs to both left and right:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone='both')
```

```
>>> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and accept LilyPond default nibs at both left and right:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone=True)

>>> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Do not beam lone leaf:

```
>>> note = Note("c'16")

>>> beam = beamtools.ComplexBeamSpanner([note], lone=False)

>>> f(note)
c'16
```

Set to 'left', 'right', 'both', true or false as shown above.

Ignore this setting when spanner contains more than one leaf.

Inherited from `beamtools.ComplexBeamSpanner`

`MeasuredComplexBeamSpanner`. **span**

Get top-level beam count:

```
>>> staff = Staff([Measure((2, 16), "c'16 d'16"), Measure((2, 16), "e'16 f'16")])
>>> beam = beamtools.MeasuredComplexBeamSpanner(staff.leaves)
>>> beam.span
1
```

Set top-level beam count:

```
>>> staff = Staff([Measure((2, 16), "c'16 d'16"), Measure((2, 16), "e'16 f'16")])
>>> beam = beamtools.MeasuredComplexBeamSpanner(staff.leaves)
>>> beam.span = 2
>>> beam.span
2
```

Set nonnegative integer.

Methods

`MeasuredComplexBeamSpanner`. **append** (*component*)

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.append_left` (*component*)

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.clear` ()

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.extend` (*components*)

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
  c'8 [
```

```

        d'8
        e'8
        f'8 ]
    }

```

Return list.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0

```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.pop()`

Remove and return rightmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)

```

Return component.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.pop_left()`

Remove and return leftmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)

```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`MeasuredComplexBeamSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__len__()`

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

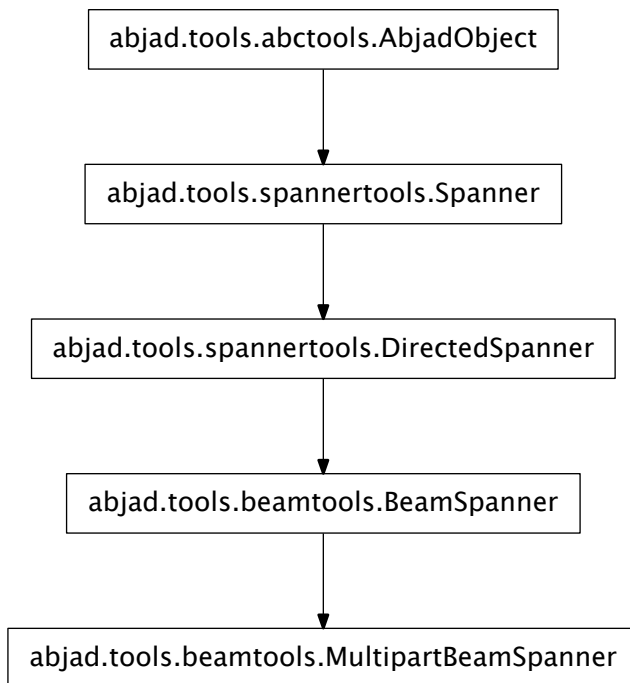
Return boolean.

Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__repr__()`

Inherited from `spannertools.Spanner`

beamtools.MultipartBeamSpanner



class `beamtools.MultipartBeamSpanner` (*components=None, direction=None*)

New in version 2.0. Abjad multipart beam spanner:

```

>>> staff = Staff("c'8 d'8 e'4 f'8 g'8 r4")

>>> beamtools.MultipartBeamSpanner(staff[:])
MultipartBeamSpanner(c'8, d'8, e'4, f'8, g'8, r4)

>>> f(staff)
\new Staff {
    c'8 [
    d'8 ]
    e'4
    f'8 [
    g'8 ]
    
```

```
    r4
}
```

Avoid rests.

Avoid large-duration notes.

Return multipart beam spanner.

Read-only Properties

`MultipartBeamSpanner.components`

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.stop_offset`
Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.written_duration`
Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`MultipartBeamSpanner.direction`
Inherited from `spannertools.DirectedSpanner`

Methods

`MultipartBeamSpanner.append(component)`
Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.append_left(component)`
Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.clear()`
Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`MultipartBeamSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__contains__(expr)`
 Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__getitem__(expr)`
 Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

functions

`beamtools.apply_beam_spanners_to_measures_in_expr`

`beamtools.apply_beam_spanners_to_measures_in_expr(expr)`
 New in version 1.1. Apply beam spanners to measures in *expr*:

```
>>> staff = Staff(r"abj: | 2/8 c'8 d'8 || 2/8 e'8 f'8 |")
```

```
>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
}
```

```
>>> beamtools.apply_beam_spanners_to_measures_in_expr(staff)
[BeamSpanner(|2/8(2)|), BeamSpanner(|2/8(2)|)]
```

```
>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 [
    d'8 ]
  }
  {
    e'8 [
    f'8 ]
  }
}
```

Return list of beams created. Changed in version 2.0: renamed `measuretools.beam()` to `beamtools.apply_beam_spanners_to_measures_in_expr()`. Changed in version 2.9: renamed `measuretools.apply_beam_spanners_to_measures_in_expr()` to `beamtools.apply_beam_spanners_to_measures_in_expr()`.

beamtools.apply_complex_beam_spanners_to_measures_in_expr

beamtools.apply_complex_beam_spanners_to_measures_in_expr(*expr*)

New in version 2.0. Apply complex beam spanners to measures in *expr*:

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
```

```
>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
}
```

```
>>> beamtools.apply_complex_beam_spanners_to_measures_in_expr(staff)
[ComplexBeamSpanner(|2/8(2)|), ComplexBeamSpanner(|2/8(2)|)]
```

```
>>> f(staff)
\new Staff {
  {
    \time 2/8
    \set stemLeftBeamCount = #0
    \set stemRightBeamCount = #1
    c'8 [
    \set stemLeftBeamCount = #1
    \set stemRightBeamCount = #0
    d'8 ]
  }
  {
    \set stemLeftBeamCount = #0
    \set stemRightBeamCount = #1
    e'8 [
    \set stemLeftBeamCount = #1
    \set stemRightBeamCount = #0
    f'8 ]
  }
}
```

Return list of beams created. Changed in version 2.9: renamed
`measuretools.apply_complex_beam_spanners_to_measures_in_expr()` to
`beamtools.apply_complex_beam_spanners_to_measures_in_expr()`.

beamtools.apply_durated_complex_beam_spanner_to_measures

`beamtools.apply_durated_complex_beam_spanner_to_measures` (*measures*)

New in version 1.1. Apply durated complex beam spanner to *measures*:

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
}

>>> measures = staff[:]
>>> beamtools.apply_durated_complex_beam_spanner_to_measures(measures)
DuratedComplexBeamSpanner(|2/8(2)|, |2/8(2)|)

>>> f(staff)
\new Staff {
  {
    \time 2/8
```

```

        \set stemLeftBeamCount = #0
        \set stemRightBeamCount = #1
        c'8 [
        \set stemLeftBeamCount = #1
        \set stemRightBeamCount = #1
        d'8
    ]
}
{
    \set stemLeftBeamCount = #1
    \set stemRightBeamCount = #1
    e'8
    \set stemLeftBeamCount = #1
    \set stemRightBeamCount = #0
    f'8 ]
}
}

```

Set beam spanner durations to preprolated measure durations.

Return beam spanner created.	Changed in version 2.0:	renamed
<code>measuretools.beam_together()</code>	Changed in version 2.9:	renamed
<code>measuretools.apply_durated_complex_beam_spanner_to_measures()</code>		to
<code>beamtools.apply_durated_complex_beam_spanner_to_measures()</code>		

beamtools.apply_multipart_beam_spanner_to_bottommost_tuplets_in_expr

`beamtools.apply_multipart_beam_spanner_to_bottommost_tuplets_in_expr(expr)`

Beam bottommost tuplets in *expr*:

```

>>> staff = Staff(3 * Tuplet(Fraction(2, 3), "c'8 d'8 e'8"))

f(staff)
\new Staff {
    \times 2/3 {
        c'8
        d'8
        e'8
    }
    \times 2/3 {
        c'8
        d'8
        e'8
    }
    \times 2/3 {
        c'8
        d'8
        e'8
    }
}

>>> beamtools.apply_multipart_beam_spanner_to_bottommost_tuplets_in_expr(staff)

>>> f(staff)
\new Staff {
    \times 2/3 {
        c'8 [
        d'8

```



```

        e'8 ]
    }
    \times 2/3 {
        c'8 [
        d'8
        e'8 ]
    }
    \times 2/3 {
        c'8 [
        d'8
        e'8 ]
    }
}

```

Return none. Changed in version 2.9: renamed `tuplettools.beam_bottommost_tuplets_in_expr()` to `beamtools.apply_multipart_beam_spanner_to_bottommost_tuplets_in_expr()`. Changed in version 2.9: renamed `beamtools.beam_bottommost_tuplets_in_expr()` to `beamtools.apply_multipart_beam_spanner_to_bottommost_tuplets_in_expr()`.

beamtools.get_beam_spanner_attached_to_component

`beamtools.get_beam_spanner_attached_to_component` (*component*)

New in version 2.0. Get the only beam spanner attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)

>>> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

>>> beamtools.get_beam_spanner_attached_to_component(staff[0])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> _ is beam
True

```

Return beam spanner.

Raise missing spanner error when no beam spanner attached to *component*.

Raise extra spanner error when more than one beam spanner attached to *component*. Changed in version 2.9: renamed `spannertools.get_beam_spanner_attached_to_component()` to `beamtools.get_beam_spanner_attached_to_component()`.

beamtools.is_beamable_component

`beamtools.is_beamable_component` (*expr*)

New in version 1.1. True when *expr* is a beamable component. Otherwise false:

```

>>> beamtools.is_beamable_component(Note(13, (1, 16)))
True

```

Return boolean. Changed in version 2.9: renamed `componenttools.is_beamable_component()` to `beamtools.is_beamable_component()`.

beamtools.is_component_with_beam_spanner_attached

`beamtools.is_component_with_beam_spanner_attached(expr)`

New in version 2.0. True when *expr* is component with beam spanner attached:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)

>>> beamtools.is_component_with_beam_spanner_attached(staff[0])
True
```

Otherwise false:

```
>>> note = Note("c'8")

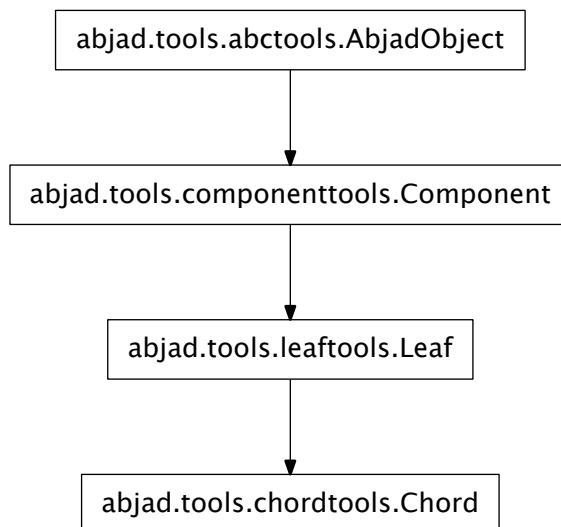
>>> beamtools.is_component_with_beam_spanner_attached(note)
False
```

Return boolean. Changed in version 2.9: renamed `spannertools.is_component_with_beam_spanner_attached()` to `beamtools.is_component_with_beam_spanner_attached()`.

chordtools

concrete classes

chordtools.Chord



class chordtools.**Chord** (*args, **kwargs)
 Abjad model of a chord:

```
>>> Chord([4, 13, 17], (1, 4))
Chord("<e' cs'' f''>4")
```

Return chord instance.

Read-only Properties

Chord.duration_in_seconds

Inherited from leaftools.Leaf

Chord.fingered_pitches

Read-only fingered pitches:

```
>>> staff = Staff("<c'' e''>4 <d'' fs''>4")
>>> glockenspiel = instrumenttools.Glockenspiel()(staff)
>>> instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch(staff)

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Glockenspiel }
  \set Staff.shortInstrumentName = \markup { Gkspl. }
  <c' e'>4
  <d' fs'>4
}

>>> staff[0].fingered_pitches
(NamedChromaticPitch("c'"), NamedChromaticPitch("e'"))
```

Return tuple of named chromatic pitches.

Chord.leaf_index

Inherited from leaftools.Leaf

Chord.lilypond_format

Inherited from componenttools.Component

Chord.multiplied_duration

Inherited from leaftools.Leaf

Chord.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from componenttools.Component

Chord.parent

Inherited from componenttools.Component

Chord.preprolated_duration

Inherited from leaftools.Leaf

Chord.prolated_duration

Inherited from componenttools.Component

Chord.prolation

Inherited from componenttools.Component

Chord.set

Read-only reference LilyPond context setting component plug-in.

Inherited from componenttools.Component

Chord.sounding_pitches

Read-only sounding pitches:

```
>>> staff = Staff("<c' e'>4 <d' fs'>4")
>>> glockenspiel = instrumenttools.Glockenspiel()(staff)
>>> instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch(staff)

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Glockenspiel }
  \set Staff.shortInstrumentName = \markup { Gkspl. }
  <c' e'>4
  <d' fs'>4
}

>>> staff[0].sounding_pitches
(NamedChromaticPitch("c'"), NamedChromaticPitch("e'"))
```

Return tuple of named chromatic pitches.

Chord.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Chord.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

Chord.start_offset_in_seconds

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

Chord.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

Chord.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Chord.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Chord.duration_multiplier

Inherited from `leaftools.Leaf`

Chord.note_heads

Get read-only tuple of note heads in chord:

```
>>> chord = Chord([7, 12, 16], (1, 4))
>>> chord.note_heads
(NoteHead("g'"), NoteHead("c'"), NoteHead("e'"))
```

Set chord note heads from any iterable:

```
>>> chord = Chord([7, 12, 16], (1, 4))
>>> chord.note_heads = [0, 2, 6]
>>> chord
Chord("<c' d' fs'>4")
```

Chord.written_duration

Inherited from `leaftools.Leaf`

Chord.written_pitch_indication_is_at_sounding_pitch

Inherited from `leaftools.Leaf`

Chord.written_pitch_indication_is_nonsemantic

Inherited from `leaftools.Leaf`

Chord.written_pitches

Get read-only tuple of pitches in chord:

```
>>> chord = Chord([7, 12, 16], (1, 4))
>>> chord.written_pitches
(NamedChromaticPitch("g'"), NamedChromaticPitch("c'"), NamedChromaticPitch("e'"))
```

Set chord pitches from any iterable:

```
>>> chord = Chord([7, 12, 16], (1, 4))
>>> chord.written_pitches = [0, 2, 6]
>>> chord
Chord("<c' d' fs'>4")
```

Methods

Chord.append(*note_head_token*)

Append *note_head_token* to chord:

```
>>> chord = Chord([4, 13, 17], (1, 4))
>>> chord
Chord("<e' cs' f'>4")

>>> chord.append(19)
>>> chord
Chord("<e' cs' f' g'>4")
```

Sort chord note heads automatically after append and return none.

Chord.clear()

Clear chord:

```
>>> chord = Chord("<e' cs' f'>4")
>>> chord
Chord("<e' cs' f'>4")

>>> chord.clear()
>>> chord
Chord('<>4')
```

Return none.

Chord.extend(*note_head_tokens*)

Extend chord with *note_head_tokens*:

```
>>> chord = Chord([4, 13, 17], (1, 4))
>>> chord
Chord("<e' cs' ' f' '>4")

>>> chord.extend([2, 12, 18])
>>> chord
Chord("<d' e' c' ' cs' ' f' ' fs' '>4")
```

Sort chord note heads automatically after extend and return none.

`Chord.pop` (*i*=-1)

Remove note head at index *i* in chord:

```
>>> chord = Chord([4, 13, 17], (1, 4))
>>> chord
Chord("<e' cs' ' f' '>4")

>>> chord.pop(1)
NoteHead("cs' ")

>>> chord
Chord("<e' f' '>4")
```

Return note head.

`Chord.remove` (*note_head*)

Remove *note_head* from chord:

```
>>> chord = Chord([4, 13, 17], (1, 4))
>>> chord
Chord("<e' cs' ' f' '>4")

>>> chord.remove(chord[1])
>>> chord
Chord("<e' f' '>4")
```

Return none.

Special Methods

`Chord.__and__` (*arg*)

Inherited from `leaftools.Leaf`

`Chord.__contains__` (*arg*)

`Chord.__delitem__` (*i*)

`Chord.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Chord.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Chord.__getitem__` (*i*)

`Chord.__gt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Chord.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Chord.__len__()`

`Chord.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Chord.__mul__(n)`
 Inherited from `componenttools.Component`

`Chord.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Chord.__or__(arg)`
 Inherited from `leaftools.Leaf`

`Chord.__repr__()`
 Inherited from `leaftools.Leaf`

`Chord.__rmul__(n)`
 Inherited from `componenttools.Component`

`Chord.__setitem__(i, arg)`

`Chord.__str__()`
 Inherited from `leaftools.Leaf`

`Chord.__sub__(arg)`
 Inherited from `leaftools.Leaf`

`Chord.__xor__(arg)`
 Inherited from `leaftools.Leaf`

functions

`chordtools.all_are_chords`

`chordtools.all_are_chords(expr)`
 New in version 2.6. True when *expr* is a sequence of Abjad chords:

```
>>> chords = [Chord("<c' e' g'>4"), Chord("<c' f' a'>4")]
```

```
>>> chordtools.all_are_chords(chords)
True
```

True when *expr* is an empty sequence:

```
>>> chordtools.all_are_chords([])
True
```

Otherwise false:

```
>>> chordtools.all_are_chords('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

chordtools.arpeggiate_chord

`chordtools.arpeggiate_chord(chord)`

New in version 1.1. Arpeggiate *chord*:

```
>>> chord = Chord("<c' d' ' ef''>8")

>>> chordtools.arpeggiate_chord(chord)
[Note("c'8"), Note("d''8"), Note("ef''8")]
```

Arpeggiated notes inherit *chord* written duration.

Arpeggiated notes do not inherit other *chord* attributes.

Return list of newly constructed notes. Changed in version 2.0: renamed `chordtools.arpeggiate()` to `chordtools.arpeggiate_chord()`.

chordtools.change_defective_chord_to_note_or_rest

`chordtools.change_defective_chord_to_note_or_rest(chord)`

New in version 1.1. Change zero-length *chord* to rest:

```
>>> chord = Chord([], (3, 16))

>>> chord
Chord('<>8.')
```

```
>>> chordtools.change_defective_chord_to_note_or_rest(chord)
Rest('r8.')
```

Change length-one chord to note:

```
>>> chord = Chord("<cs''>8.")

>>> chord
Chord("<cs''>8.")

>>> chordtools.change_defective_chord_to_note_or_rest(chord)
Note("cs''8.")
```

Return chords with length greater than one unchanged:


```
>>> chord = Chord("<c' c' ' cs' '>8.")
>>> chord
Chord("<c' c' ' cs' '>8.")
>>> chordtools.change_defective_chord_to_note_or_rest(chord)
Chord("<c' c' ' cs' '>8.")
```

Return notes unchanged:

```
>>> note = Note("c' 4")
>>> note
Note("c' 4")
>>> chordtools.change_defective_chord_to_note_or_rest(note)
Note("c' 4")
```

Return rests unchanged:

```
>>> rest = Rest('r4')
>>> rest
Rest('r4')
>>> chordtools.change_defective_chord_to_note_or_rest(rest)
Rest('r4')
```

Return note, rest, chord or none. Changed in version 2.0: renamed `chordtools.cast_defective()` to `chordtools.change_defective_chord_to_note_or_rest()`.

chordtools.divide_chord_by_chromatic_pitch_number

`chordtools.divide_chord_by_chromatic_pitch_number(chord,`
`pitch=NamedChromaticPitch('b')`

New in version 1.1. Divide *chord* by chromatic *pitch* number:

```
>>> chord = Chord(range(12), Duration(1, 4))
>>> chord
Chord("<c' cs' d' ef' e' f' fs' g' af' a' bf' b'>4")
>>> chordtools.divide_chord_by_chromatic_pitch_number(chord, pitchtools.NamedChromaticPitch(6))
(Chord("<fs' g' af' a' bf' b'>4"), Chord("<c' cs' d' ef' e' f'>4"))
```

Input *chord* may be a note, rest or chord but not a skip.

Zero-length parts return rests, length-one parts return notes and other parts return chords.

Return pair of newly constructed leaves. Changed in version 2.0: renamed `chordtools.split_by_pitch_number()` to `chordtools.divide_chord_by_chromatic_pitch_number()`.

chordtools.divide_chord_by_diatonic_pitch_number

`chordtools.divide_chord_by_diatonic_pitch_number(chord,`
`pitch=NamedChromaticPitch('b')`

New in version 1.1. Divide *chord* by diatonic *pitch* number:

```
>>> chord = Chord(range(12), Duration(1, 4))

>>> chord
Chord("<c' cs' d' ef' e' f' fs' g' af' a' bf' b'>4")

>>> chordtools.divide_chord_by_diatonic_pitch_number(chord, pitchtools.NamedChromaticPitch(6))
(Chord("<f' fs' g' af' a' bf' b'>4"), Chord("<c' cs' d' ef' e'>4"))
```

Input *chord* may be a note, rest or chord but not a skip.

Zero-length parts return as rests, length-one parts return as notes and other parts return as chords.

Return pair of newly constructed leaves. Changed in version 2.0: renamed `chordtools.split_by_altitude()` to `chordtools.divide_chord_by_diatonic_pitch_number()`.

`chordtools.get_arithmetic_mean_of_chord`

`chordtools.get_arithmetic_mean_of_chord(chord)`

New in version 2.0. Get arithmetic mean of chromatic pitch number of pitches in *chord*:

```
>>> chord = Chord("<g' c' ' e' ' >4")

>>> chordtools.get_arithmetic_mean_of_chord(chord)
11.666666666666666
```

Return none when *chord* is empty:

```
>>> chord = Chord("< >4")

>>> chordtools.get_arithmetic_mean_of_chord(chord) is None
True
```

Return number or none.

`chordtools.get_note_head_from_chord_by_pitch`

`chordtools.get_note_head_from_chord_by_pitch(chord, pitch)`

New in version 2.0. Get note head from *chord* by *pitch*:

```
>>> chord = Chord("<c' ' d' ' b' ' >4")

>>> chordtools.get_note_head_from_chord_by_pitch(chord, 14)
NoteHead("d' ")
```

Raise missing note head error when *chord* contains no note head with pitch equal to *pitch*.

Raise extra note head error when *chord* contains more than one note head with pitch equal to *pitch*. Changed in version 2.0: renamed `chordtools.get_note_head()` to `chordtools.get_note_head_from_chord_by_pitch()`.

`chordtools.make_tied_chord`

`chordtools.make_tied_chord(pitches, duration, big_endian=True)`

Returns a list of chords to fill the given duration.

Chords returned are tie spanned.

chordtools.yield_all_subchords_of_chord**chordtools.yield_all_subchords_of_chord**(*chord*)New in version 2.0. Yield all subchords of *chord* in binary string order:

```

>>> chord = Chord("<c' d' af' a'>4")

>>> for subchord in chordtools.yield_all_subchords_of_chord(chord):
...     subchord
...
Rest('r4')
Note("c'4")
Note("d'4")
Chord("<c' d'>4")
Note("af'4")
Chord("<c' af'>4")
Chord("<d' af'>4")
Chord("<c' d' af'>4")
Note("a'4")
Chord("<c' a'>4")
Chord("<d' a'>4")
Chord("<c' d' a'>4")
Chord("<af' a'>4")
Chord("<c' af' a'>4")
Chord("<d' af' a'>4")
Chord("<c' d' af' a'>4")

```

Include empty chord as rest.

Return generator of newly constructed leaves. Changed in version 2.0: renamed `chordtools.subchords()` to `chordtools.yield_all_subchords_of_chord()`.

chordtools.yield_groups_of_chords_in_sequence**chordtools.yield_groups_of_chords_in_sequence**(*sequence*)New in version 2.0. Yield groups of chords in *sequence*:

```

>>> staff = Staff("c'8 d'8 r8 r8 <e' g'>8 <f' a'>8 g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

>>> f(staff)
\new Staff {
    c'8
    d'8
    r8
    r8
    <e' g'>8
    <f' a'>8
    g'8
    a'8
    r8
    r8
    <b' d''>8
    <c'' e''>8
}

>>> for chord in chordtools.yield_groups_of_chords_in_sequence(staff):
...     chord

```

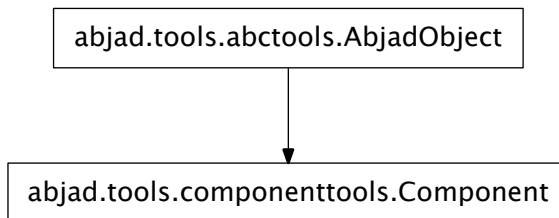
```
...
(Chord("<e' g'>8"), Chord("<f' a'>8"))
(Chord("<b' d'>8"), Chord("<c' ' e'>8"))
```

Return generator.

componenttools

concrete classes

componenttools.Component



class componenttools.Component

Read-only Properties

Component.**lilypond_format**

Component.**override**

Read-only reference to LilyPond grob override component plug-in.

Component.**parent**

Component.**prolated_duration**

Component.**prolation**

Component.**set**

Read-only reference LilyPond context setting component plug-in.

Component.**spanners**

Read-only reference to unordered set of spanners attached to component.

Component.**start_offset**

Read-only start offset of component.

Component.**start_offset_in_seconds**

Read-only start offset of comonent in seconds.

Component.**stop_offset**

Read-only stop offset of component.

Component.**stop_offset_in_seconds**

Read-only stop offset of component in seconds.

`Component.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`Component.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Component.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Component.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Component.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Component.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Component.__mul__(n)`

`Component.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Component.__repr__()`

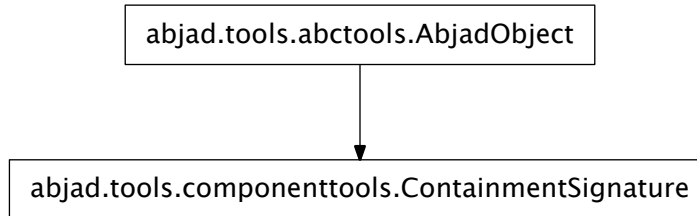
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`Component.__rmul__(n)`

componenttools.ContainmentSignature



class componenttools.ContainmentSignature

New in version 2.9. Containment signature of Abjad component:

```

>>> score = Score(r"""\context Staff = "CustomStaff" { ""
...     r"""\context Voice = "CustomVoice" { c' d' e' f' } }""")
>>> score.name = 'CustomScore'

>>> f(score)
\context Score = "CustomScore" <<
  \context Staff = "CustomStaff" {
    \context Voice = "CustomVoice" {
      c' 4
      d' 4
      e' 4
      f' 4
    }
  }
>>

>>> componenttools.component_to_containment_signature(score.leaves[0])
ContainmentSignature(Note-..., Voice-'CustomVoice', Staff-..., Score-'CustomScore')
  
```

Returned only by `componenttools.component_to_containment_signature()`.

Used for thread iteration behind the scenes.

Read-only Properties

`ContainmentSignature.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ContainmentSignature.__eq__(arg)`

`ContainmentSignature.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ContainmentSignature.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ContainmentSignature.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ContainmentSignature.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ContainmentSignature.__ne__(arg)`

`ContainmentSignature.__repr__()`

`ContainmentSignature.__str__()`

functions

`componenttools.all_are_components`

`componenttools.all_are_components(expr, classes=None)`

New in version 1.1. True when elements in *expr* are all components:

```
>>> componenttools.all_are_components(3 * Note("c'4"))
True
```

Otherwise false:

```
>>> componenttools.all_are_components(['foo', 'bar'])
False
```

True when elements in *expr* are all *klass*es:

```
>>> componenttools.all_are_components(3 * Note("c'4"), classes = Note)
True
```

Otherwise false:

```
>>> componenttools.all_are_components(['foo', 'bar'], classes = Note)
False
```

Return boolean.

`componenttools.all_are_components_in_same_parent`

`componenttools.all_are_components_in_same_parent(expr, classes=None, al-
low_orphans=True)`

New in version 1.1. True when elements in *expr* are all components in same parent. Otherwise false:

```
>>> staff = Staff(notetools.make_notes([12, 14, 16], [(1, 8)]))
>>> componenttools.all_are_components_in_same_parent(staff.leaves)
True
```

True when elements in *expr* are all *klases* in same parent. Otherwise false:

```
>>> staff = Staff(notetools.make_notes([12, 14, 16], [(1, 8)]))
>>> componenttools.all_are_components_in_same_parent(staff.leaves, classes=(Note, ))
True
```

Return boolean.

componenttools.all_are_components_in_same_score

`componenttools.all_are_components_in_same_score`(*expr*, *klases=None*, *allow_orphans=True*)

New in version 1.1. True when elements in *expr* are all components in same score. Otherwise false:

```
>>> score = Score([Staff("c'8 d'8 e'8")])
>>> componenttools.all_are_components_in_same_score(score.leaves)
True
```

True when elements in *expr* are all *klases* in same score. Otherwise false:

```
>>> score = Score([Staff("c'8 d'8 e'8")])
>>> componenttools.all_are_components_in_same_score(score.leaves, classes=(Note, ))
True
```

Return boolean.

componenttools.all_are_components_in_same_thread

`componenttools.all_are_components_in_same_thread`(*expr*, *klases=None*, *allow_orphans=True*)

New in version 1.1. True when elements in *expr* are all components in same thread. Otherwise false:

```
>>> voice = Voice("c'8 d'8 e'8")
>>> componenttools.all_are_components_in_same_thread(voice.leaves)
True
```

True when elements in *expr* are all *klases* in same thread. Otherwise false:

```
>>> voice = Voice("c'8 d'8 e'8")
>>> componenttools.all_are_components_in_same_thread(voice.leaves, klases=Note)
True
```

Return boolean.

componenttools.all_are_components_scalable_by_multiplier

`componenttools.all_are_components_scalable_by_multiplier`(*components*, *multiplier*)

New in version 1.1. True when *components* are all scalable by *multiplier*:

```
>>> components = [Note(0, (1, 8))]
>>> componenttools.all_are_components_scalable_by_multiplier(components, Duration(3, 2))
True
```


Otherwise false:

```
>>> components = [Note(0, (1, 8))]
>>> componenttools.all_are_components_scalable_by_multiplier(components, Duration(2, 3))
False
```

Return boolean. Changed in version 2.0: renamed `durationtools.are_scalable()` to `componenttools.all_are_components_scalable_by_multiplier()`.

componenttools.all_are_contiguous_components

`componenttools.all_are_contiguous_components` (*expr*, *classes=None*, *allow_orphans=True*)

New in version 1.1. True when elements in *expr* are all contiguous components. Otherwise false:

```
>>> staff = Staff("c'8 d'8 e'8")
>>> componenttools.all_are_contiguous_components(staff.leaves)
True
```

True when elements in *expr* are all contiguous *classes*. Otherwise false:

```
>>> staff = Staff("c'8 d'8 e'8")
>>> componenttools.all_are_contiguous_components(staff.leaves, classes=Note)
True
```

Return boolean.

componenttools.all_are_contiguous_components_in_same_parent

`componenttools.all_are_contiguous_components_in_same_parent` (*expr*,
classes=None, *allow_orphans=True*)

New in version 1.1. True when elements in *expr* are all contiguous components in same parent. Otherwise false:

```
>>> staff = Staff("c'8 d'8 e'8")
>>> componenttools.all_are_contiguous_components_in_same_parent(staff.leaves)
True
```

True when elements in *expr* are all contiguous *classes* in same parent. Otherwise false:

```
>>> staff = Staff("c'8 d'8 e'8")
>>> componenttools.all_are_contiguous_components_in_same_parent(staff.leaves, classes=Note)
True
```

Return boolean.

componenttools.all_are_contiguous_components_in_same_score

`componenttools.all_are_contiguous_components_in_same_score` (*expr*,
classes=None, *allow_orphans=True*)

New in version 1.1. True when elements in *expr* are all contiguous components in same score. Otherwise false:

```
>>> score = Score([Staff("c'8 d'8 e'8")])
>>> componenttools.all_are_contiguous_components_in_same_score(score.leaves)
True
```

True when elements in *expr* are all contiguous *classes* in same score. Otherwise false:

```
>>> score = Score([Staff("c'8 d'8 e'8")])
>>> componenttools.all_are_contiguous_components_in_same_score(score.leaves, classes=Note)
True
```

Return boolean.

componenttools.all_are_contiguous_components_in_same_thread

```
componenttools.all_are_contiguous_components_in_same_thread(expr,
                                                            classes=None, al-
                                                            low_orphans=True)
```

New in version 1.1. True when elements in *expr* are all contiguous components in same thread. Otherwise false:

```
>>> staff = Staff("c'8 d'8 e'8")
>>> componenttools.all_are_contiguous_components_in_same_thread(staff.leaves)
True
```

True when elements in *expr* are all contiguous *classes* in same thread. Otherwise false:

```
>>> staff = Staff("c'8 d'8 e'8")
>>> componenttools.all_are_contiguous_components_in_same_thread(staff.leaves, classes=Note)
True
```

Return boolean.

componenttools.all_are_orphan_components

```
componenttools.all_are_orphan_components(expr)
```

New in version 2.0. True when *expr* is an iterable of zero or more orphan components.

Otherwise false.

componenttools.all_are_thread_contiguous_components

```
componenttools.all_are_thread_contiguous_components(expr, classes=None, al-
                                                    low_orphans=True)
```

New in version 1.1. True when elements in *expr* are all thread-contiguous components:

```
t = Voice(notetools.make_repeated_notes(4))
t.insert(2, Voice(notetools.make_repeated_notes(2)))
Container(t[:2])
Container(t[-2:])
pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(t)
```

```
\new Voice {
  {
    c'8
    d'8
  }
  \new Voice {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
}
```

```

    }
}

assert _are_thread_contiguous_components(t[0:1] + t[-1:])
assert _are_thread_contiguous_components(t[0][:] + t[-1:])
assert _are_thread_contiguous_components(t[0:1] + t[-1][:])
assert _are_thread_contiguous_components(t[0][:] + t[-1][:])

```

Return boolean.

Thread-contiguous components are, by definition, spannable.

componenttools.component_to_containment_signature

`componenttools.component_to_containment_signature(component)`

New in version 1.1. Change *component* to containment signature:

```

>>> score = Score(
...     r"""\context Staff = "CustomStaff" { ""
...     r"""\context Voice = "CustomVoice" { c' d' e' f' } """
>>> score.name = 'CustomScore'

>>> f(score)
\context Score = "CustomScore" <<
  \context Staff = "CustomStaff" {
    \context Voice = "CustomVoice" {
      c' 4
      d' 4
      e' 4
      f' 4
    }
  }
>>

>>> componenttools.component_to_containment_signature(score.leaves[0])
ContainmentSignature(Note-..., Voice-'CustomVoice', Staff-..., Score-'CustomScore')

```

Containment signature gives first voice, staff, staff group and score in parentage. Changed in version 2.9: renamed `threadtools.component_to_thread_signature()` to `componenttools.component_to_containment_signature()`.

componenttools.component_to_parentage_signature

`componenttools.component_to_parentage_signature(component)`

New in version 1.1. Change *component* to parentage signature:

```

>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> staff = Staff([tuplet])
>>> note = staff.leaves[0]
>>> print componenttools.component_to_parentage_signature(note)
staff: Staff-...
self: Note-...

```

Return parentage signature.

componenttools.component_to_pitch_and_rhythm_skeleton

`componenttools.component_to_pitch_and_rhythm_skeleton(component)`

New in version 2.0. Change *component* to pitch and rhythm skeleton:

```
>>> tuplet = Tuplet(Fraction(3, 4), "c'8 d'8 e'8 f'8")
>>> measure = Measure((6, 16), [tuplet])
>>> staff = Staff([measure])
>>> score = Score(staff * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(score)
```

```
>>> skeleton = componenttools.component_to_pitch_and_rhythm_skeleton(score)
>>> print skeleton
```

```
Score([
  Staff([
    Measure((6, 16), [
      Tuplet(Fraction(3, 4), [
        Note(('c', 4), Duration(1, 8)),
        Note(('d', 4), Duration(1, 8)),
        Note(('e', 4), Duration(1, 8)),
        Note(('f', 4), Duration(1, 8))
      ])
    ])
  ],
  Staff([
    Measure((6, 16), [
      Tuplet(Fraction(3, 4), [
        Note(('g', 4), Duration(1, 8)),
        Note(('a', 4), Duration(1, 8)),
        Note(('b', 4), Duration(1, 8)),
        Note(('c', 5), Duration(1, 8))
      ])
    ])
  ])
])
```

```
>>> new = eval(skeleton)
>>> new
Score<<2>>
```

```
>>> f(new)
\new Score <<
  \new Staff {
    {
      \time 6/16
      \fraction \times 3/4 {
        c'8
        d'8
        e'8
        f'8
      }
    }
  }
  \new Staff {
    {
      \time 6/16
      \fraction \times 3/4 {
        g'8
```

```

        a'8
        b'8
        c''8
    }
}
>>

```

Return string.

componenttools.component_to_score_depth

componenttools.component_to_score_depth(component)

New in version 1.1. Change *component* to score depth:

```

>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> staff = Staff([tuplet])
>>> componenttools.component_to_score_depth(staff.leaves[0])
2

```

Return nonnegative integer.

componenttools.component_to_score_index

componenttools.component_to_score_index(component)

New in version 2.0. Change *component* to score index:

```

>>> staff_1 = Staff(r"\times 2/3 { c'8 d'8 e'8 } \times 2/3 { f'8 g'8 a'8 }")
>>> staff_2 = Staff(r"\times 2/3 { b'8 c''8 d''8 }")
>>> score = Score([staff_1, staff_2])

```

```

>>> f(score)
\nnew Score <<
  \new Staff {
    \times 2/3 {
      c'8
      d'8
      e'8
    }
    \times 2/3 {
      f'8
      g'8
      a'8
    }
  }
  \new Staff {
    \times 2/3 {
      b'8
      c''8
      d''8
    }
  }
>>

>>> for leaf in score.leaves:
...     leaf, componenttools.component_to_score_index(leaf)

```

```
...
(Note("c'8"), (0, 0, 0))
(Note("d'8"), (0, 0, 1))
(Note("e'8"), (0, 0, 2))
(Note("f'8"), (0, 1, 0))
(Note("g'8"), (0, 1, 1))
(Note("a'8"), (0, 1, 2))
(Note("b'8"), (1, 0, 0))
(Note("c''8"), (1, 0, 1))
(Note("d''8"), (1, 0, 2))
```

Return tuple of zero or more nonnegative integers.

componenttools.component_to_score_root

`componenttools.component_to_score_root` (*component*)

New in version 1.1. Change *component* to score root:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> staff = Staff([tuplet])
>>> note = staff.leaves[0]
>>> componenttools.component_to_score_root(note)
Staff{1}
```

Return score root.

componenttools.component_to_tuplet_depth

`componenttools.component_to_tuplet_depth` (*component*)

New in version 1.1. Change *component* to tuplet depth:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> staff = Staff([tuplet])
>>> note = staff.leaves[0]

>>> componenttools.component_to_tuplet_depth(note)
1

>>> componenttools.component_to_tuplet_depth(tuplet)
0

>>> componenttools.component_to_tuplet_depth(staff)
0
```

Return nonnegative integer.

componenttools.copy_and_partition_governed_component_subtree_by_leaf_counts

`componenttools.copy_and_partition_governed_component_subtree_by_leaf_counts` (*container*,
leaf_counts)

New in version 1.1. Copy *container* and partition copy according to *leaf_counts*:

```
>>> voice = Voice(r"\times 2/3 { c'8 d'8 e'8 } \times 2/3 { f'8 g'8 a'8 }")
>>> beamtools.BeamSpanner(voice[0].leaves)
BeamSpanner(c'8, d'8, e'8)
```

```

>>> beamtools.BeamSpanner(voice[1].leaves)
BeamSpanner(f'8, g'8, a'8)

>>> f(voice)
\new Voice {
  \times 2/3 {
    c'8 [
      d'8
      e'8 ]
  }
  \times 2/3 {
    f'8 [
      g'8
      a'8 ]
  }
}

>>> result = componenttools.copy_and_partition_governed_component_subtree_by_leaf_counts(
... voice, [1, 2, 3])

>>> first, second, third = result

>>> f(first)
\new Voice {
  \times 2/3 {
    c'8 [ ]
  }
}

>>> f(second)
\new Voice {
  \times 2/3 {
    d'8 [
      e'8 ]
  }
}

>>> f(third)
\new Voice {
  \times 2/3 {
    f'8 [
      g'8
      a'8 ]
  }
}

```

Set *leaf_counts* to an iterable of zero or more positive integers.

Return a list of parts equal in length to that of *leaf_counts*. Changed in version 2.0: renamed `clonewp.by_leaf_counts_with_parentage()` to `componenttools.copy_and_partition_governed_component_subtree_by_leaf_counts()`.

componenttools.copy_components_and_covered_spanners

`componenttools.copy_components_and_covered_spanners` (*components*, *n=1*)

New in version 1.1. Copy *components* and covered spanners.

The *components* must be thread-contiguous.

Covered spanners are those spanners that cover *components*.

The steps taken in this function are as follows. Withdraw *components* from crossing spanners. Preserve spanners that *components* cover. Deep copy *components*. Reapply crossing spanners to source *components*. Return copied components with covered spanners.

```
>>> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(voice)
>>> beam = beamtools.BeamSpanner(voice.leaves[:4])
>>> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
      d'8
    ]
  }
  {
    e'8
    f'8 ]
  }
  {
    g'8
    a'8
  }
}

>>> result = componenttools.copy_components_and_covered_spanners(voice.leaves)
>>> result
(Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'8"), Note("a'8"))

>>> new_voice = Voice(result)
>>> f(new_voice)
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
  g'8
  a'8
}

>>> voice.leaves[0] is new_voice.leaves[0]
False
```

Copy *components* a total of *n* times.

```
>>> result = componenttools.copy_components_and_covered_spanners(voice.leaves[:2], n=3)
>>> result
(Note("c'8"), Note("d'8"), Note("c'8"), Note("d'8"), Note("c'8"), Note("d'8"))

>>> new_voice = Voice(result)
>>> f(new_voice)
\new Voice {
  c'8
  d'8
  c'8
  d'8
  c'8
  d'8
}
```


Changed in version 2.0: renamed `clone.covered()` to `componenttools.copy_components_and_covered_spanners()`.
 in version 2.0: renamed `componenttools.clone_components_and_covered_spanners()` to `componenttools.copy_components_and_covered_spanners()`.

`componenttools.copy_components_and_fracture_crossing_spanners`

`componenttools.copy_components_and_fracture_crossing_spanners`(*components*, *n=1*)

New in version 1.1. Copy *components* and fracture crossing spanners.

The *components* must be thread-contiguous.

The steps this function takes are as follows. Deep copy *components*. Deep copy spanners that attach to any component in *components*. Fracture spanners that attach to components not in *components*. Return Python list of copied components.

```
>>> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(voice)
>>> beam = beamtools.BeamSpanner(voice.leaves[:4])
>>> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
    d'8
  ]
  {
    e'8
    f'8 ]
  }
  {
    g'8
    a'8
  }
}

>>> result = componenttools.copy_components_and_fracture_crossing_spanners(
...     voice.leaves[2:4])
>>> result
(Note("e'8"), Note("f'8"))

>>> new_voice = Voice(result)
>>> f(new_voice)
\new Voice {
  e'8 [
  f'8 ]
}

>>> voice.leaves[2] is new_voice.leaves[0]
False
```

Copy *components* a total of *n* times.

```
>>> result = componenttools.copy_components_and_fracture_crossing_spanners(
...     voice.leaves[2:4], n=3)
>>> result
(Note("e'8"), Note("f'8"), Note("e'8"), Note("f'8"), Note("e'8"), Note("f'8"))
```

```
>>> new_voice = Voice(result)
>>> f(new_voice)
\new Voice {
  e'8 [
  f'8 ]
  e'8 [
  f'8 ]
  e'8 [
  f'8 ]
}
```

Changed in version 2.0: renamed `clone.fracture()` to `componenttools.copy_components_and_fracture_cr`
 in version 2.0: renamed `componenttools.clone_components_and_fracture_crossing_spanners()`
 to `componenttools.copy_components_and_fracture_crossing_spanners()`.

`componenttools.copy_components_and_immediate_parent_of_first_component`

`componenttools.copy_components_and_immediate_parent_of_first_component(components)`
 New in version 1.1. Copy *components* and immediate parent of first component.

The *components* must be thread-contiguous.

Return in newly created container equal to type of first element in *components*.

If the parent of the first element in *components* is a tuplet then insure that the tuplet multiplier of the function output equals the tuplet multiplier of the parent of the first element in *components*.

```
>>> voice = Voice(r"\times 2/3 { c'8 d' e' } \times 2/3 { f'8 g' a' }")
>>> voice.append(r"\times 2/3 { b'8 c'' d'' }")
>>> beam = beamtools.BeamSpanner(voice.leaves[:4])

>>> f(voice)
\new Voice {
  \times 2/3 {
    c'8 [
    d'8
    e'8
  ]
  \times 2/3 {
    f'8 ]
    g'8
    a'8
  ]
  \times 2/3 {
    b'8
    c''8
    d''8
  ]
}

>>> leaves = voice.leaves[:2]
>>> componenttools.copy_components_and_immediate_parent_of_first_component(leaves)
Tuplet(2/3, [c'8, d'8])
```

Parent-contiguity is not required. Thread-contiguous *components* suffice.

```
>>> leaves = voice.leaves[:5]
>>> componenttools.copy_components_and_immediate_parent_of_first_component(leaves)
Tuplet(2/3, [c'8, d'8, e'8, f'8, g'8])
```

Note: this function copies only the *immediate parent* of the first element in *components*. This function ignores any further parentage of *components* above the immediate parent of *components*.

Todo

this function should (but does not) copy marks that attach to *components* and to the immediate parent of the first component; extend function to do so.

Changed in version 2.0: renamed `clonewp.with_parent()` to `componenttools.copy_components_and_immediate_parent_of_first_component()`. Changed in version 2.0: renamed `componenttools.clone_components_and_immediate_parent_of_first_component()` to `componenttools.copy_components_and_immediate_parent_of_first_component()`.

componenttools.copy_components_and_remove_spanners

`componenttools.copy_components_and_remove_spanners(components, n=1)`

New in version 1.1. Copy *components* and remove all spanners.

The *components* must be thread-contiguous.

The steps taken by this function are as follows. Withdraw all components at any level in *components* from spanners. Deep copy unspanned components in *components*. Reapply spanners to all components at any level in *components*.

```
>>> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(voice)
>>> beam = beamtools.BeamSpanner(voice.leaves[:4])
>>> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
      d'8
    ]
  }
  {
    e'8
    f'8 ]
  }
  {
    g'8
    a'8
  }
}
```

```
>>> result = componenttools.copy_components_and_remove_spanners(voice.leaves[2:4])
>>> result
(Note("e'8"), Note("f'8"))

>>> new_voice = Voice(result)
>>> f(new_voice)
\new Voice {
```

```

    e'8
    f'8
}

>>> voice.leaves[2] is new_voice.leaves[0]
False

```

Copy *components* a total of *n* times.

```

>>> result = componenttools.copy_components_and_remove_spanners(voice.leaves[2:4], n=3)
>>> result
(Note("e'8"), Note("f'8"), Note("e'8"), Note("f'8"), Note("e'8"), Note("f'8"))

>>> new_voice = Voice(result)
>>> f(new_voice)
\new Voice {
    e'8
    f'8
    e'8
    f'8
    e'8
    f'8
}

```

Changed in version 2.0: renamed `componenttools.clone_components_and_remove_all_spanners()` to `componenttools.copy_components_and_remove_spanners()`. Changed in version 2.9: renamed `componenttools.copy_components_and_remove_all_spanners()` to `componenttools.copy_components_and_remove_spanners()`.

`componenttools.copy_governed_component_subtree_by_leaf_range`

```

componenttools.copy_governed_component_subtree_by_leaf_range(component,
                                                             start=0,
                                                             stop=None)

```

New in version 1.1. Copy governed *component* subtree by leaf range.

Governed subtree means *component* together with children of *component*.

Leaf range refers to the sequential parentage of *component* from *start* leaf index to *stop* leaf index:

```

>>> voice = Voice(r"\times 2/3 { c'8 d'8 e'8 } \times 2/3 { f'8 g'8 a'8 }")
>>> t = Staff([voice])

>>> f(t)
\new Staff {
  \new Voice {
    \times 2/3 {
      c'8
      d'8
      e'8
    }
    \times 2/3 {
      f'8
      g'8
      a'8
    }
  }
}

```

```
>>> u = componenttools.copy_governed_component_subtree_by_leaf_range(t, 1, 5)
>>> f(u)
\new Staff {
  \new Voice {
    \times 2/3 {
      d'8
      e'8
    }
    \times 2/3 {
      f'8
      g'8
    }
  }
}
```

Copy sequential containers in leaves' parentage up to the first parallel container in leaves' parentage.

Trim and shrink copied containers as necessary.

When *stop* is none copy all leaves from *start* forward. Changed in version 2.0: renamed `clonewp.by_leaf_range_with_parentage()` to `componenttools.copy_governed_component_subtree_by_leaf_range()`. Changed in version 2.0: renamed `componenttools.clone_governed_component_subtree_by_leaf_range()` to `componenttools.copy_governed_component_subtree_by_leaf_range()`.

`componenttools.copy_governed_component_subtree_from_offset_to`

```
componenttools.copy_governed_component_subtree_from_offset_to(component,
                                                             start=0,
                                                             stop=None)
```

New in version 1.1. Copy governed *component* subtree from *start* prolated duration to *stop* prolated duration.

Governed subtree refers to *component* together with the children of *component*:

```
>>> voice = Voice(r"c'8 d'8 \times 2/3 { e'8 f'8 g'8 }")

>>> f(voice)
\new Voice {
  c'8
  d'8
  \times 2/3 {
    e'8
    f'8
    g'8
  }
}

>>> new = componenttools.copy_governed_component_subtree_from_offset_to(
...     voice, (0, 8), (3, 8))

>>> f(new)
\new Voice {
  c'8
  d'8
  \times 2/3 {
    e'8
    f'16
  }
}
```

```
    }  
}
```

Raise contiguity error if asked to slice a parallel container.

```
>>> staff = Staff(Voice("c'8 d'8") * 2)  
>>> staff.is_parallel = True  
>>> f(staff)  
\new Staff <<  
\new Voice {  
    c'8  
    d'8  
}  
\new Voice {  
    c'8  
    d'8  
}  
>>
```

Raise contiguity error when attempting to copy fleaves from parallel container.

But note that cases with `0 = start` work correctly:

```
>>> new = componenttools.copy_governed_component_subtree_from_offset_to(  
...     voice, (0, 8), (1, 8))  
  
>>> f(new)  
\new Voice {  
    c'8  
}
```

Cases with `0 < start` do not work correctly:

```
>>> new = componenttools.copy_governed_component_subtree_from_offset_to(  
...     voice, (1, 8), (2, 8))  
  
>>> f(new)  
\new Voice {  
    c'8  
    d'8  
}
```

Create ad hoc tuplets as required:

```
>>> voice = Voice([Note("c'4")])  
>>> new = componenttools.copy_governed_component_subtree_from_offset_to(  
...     voice, 0, (1, 12))  
  
>>> f(new)  
\new Voice {  
    \times 2/3 {  
        c'8  
    }  
}
```

Function does NOT copy parentage of *component* when *component* is a leaf:

```
>>> voice = Voice([Note("c'4")])  
>>> new_leaf = componenttools.copy_governed_component_subtree_from_offset_to(  
...     voice[0], 0, (1, 8))
```

```
>>> f(new_leaf)
c'8

>>> new_leaf.parent is None
True

Return (untrimmed_copy, first_dif, second_dif).
```

componenttools.cut_component_at_prolated_duration

`componenttools.cut_component_at_prolated_duration` (*component*, *prolated_duration*)
New in version 2.0. Cut *component* at dotted *prolated_duration*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> componenttools.cut_component_at_prolated_duration(staff, Duration(1, 32))
>>> f(staff)
\new Staff {
  c'16. [
  d'8
  e'8
  f'8 ]
}
```

Cut *component* at tied *prolated_duration*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> componenttools.cut_component_at_prolated_duration(staff, Duration(3, 64))
>>> f(staff)
\new Staff {
  c'16 [ ~
  c'64
  d'8
  e'8
  f'8 ]
}
```

Cut *component* at nonbinary *prolated_duration*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> componenttools.cut_component_at_prolated_duration(staff, Duration(1, 24))
>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8 [
  ]
  d'8
  e'8
  f'8 ]
}
```

Return none.

componenttools.extend_in_parent_of_component

`componenttools.extend_in_parent_of_component` (*component*, *new_components*, *grow_spanners=True*)

New in version 2.10. Extend *new_components* in parent of *component*.

Example 1. Extend *new_component* in parent of *component*. Grow spanners:

```
>>> voice = Voice("c'8 [ d'8 e'8 ]")

>>> f(voice)
\new Voice {
  c'8 [
    d'8
    e'8 ]
}

>>> new_components = [Note("c'8"), Note("d'8"), Note("e'8")]
>>> componenttools.extend_in_parent_of_component(
...     voice.leaves[-1], new_components, grow_spanners=True)
[Note("e'8"), Note("c'8"), Note("d'8"), Note("e'8")]

>>> f(voice)
\new Voice {
  c'8 [
    d'8
    e'8
    c'8
    d'8
    e'8 ]
}
```

Example 2. Extend *new_component* in parent of *component*. Do not grow spanners:

```
>>> voice = Voice("c'8 [ d'8 e'8 ]")

>>> f(voice)
\new Voice {
  c'8 [
    d'8
    e'8 ]
}

>>> new_components = [Note("c'8"), Note("d'8"), Note("e'8")]
>>> componenttools.extend_in_parent_of_component(
...     voice.leaves[-1], new_components, grow_spanners=False)
[Note("e'8"), Note("c'8"), Note("d'8"), Note("e'8")]

>>> f(voice)
\new Voice {
  c'8 [
    d'8
    e'8 ]
  c'8
  d'8
  e'8
}
```

Return *component* and *new_components* together in newly constructed list.

componenttools.extend_left_in_parent_of_component

`componenttools.extend_left_in_parent_of_component` (*component*, *new_components*,
grow_spanners=True)

New in version 2.10. Extend *new_components* left in parent of *component*.

Example 1. Extend *new_components* left in parent of *component*. Grow spanners:

```
>>> voice = Voice("c'8 [ d'8 e'8 ]")

>>> f(voice)
\new Voice {
  c'8 [
    d'8
    e'8 ]
}

>>> notes = [Note("c'8"), Note("d'8"), Note("e'8")]
>>> componenttools.extend_left_in_parent_of_component(
...     voice[0], notes, grow_spanners=True)
[Note("c'8"), Note("d'8"), Note("e'8"), Note("c'8")]

>>> f(voice)
\new Voice {
  c'8 [
    d'8
    e'8
    c'8
    d'8
    e'8 ]
}
```

Example 2. Extend *new_components* left in parent of *component*. Do not grow spanners:

```
>>> voice = Voice("c'8 [ d'8 e'8 ]")

>>> notes = [Note("c'8"), Note("d'8"), Note("e'8")]
>>> componenttools.extend_left_in_parent_of_component(
...     voice[0], notes, grow_spanners=False)
[Note("c'8"), Note("d'8"), Note("e'8"), Note("c'8")]

>>> f(voice)
\new Voice {
  c'8
  d'8
  e'8
  c'8 [
    d'8
    e'8 ]
}
```

Return *new_components* and *component* together in newly created list.

componenttools.get_component_in_expr_with_name

`componenttools.get_component_in_expr_with_name` (*expr*, *name*)

New in version 2.10. Get component in *expr* with *name*:

```
>>> voice_1 = Voice("c'4 d'4 e'4 f'4")
>>> voice_1.name = 'Top Voice'
>>> voice_2 = Voice(r'\clef "bass" c4 d4 e4 f4')
>>> voice_2.name = 'Bottom Voice'
>>> staff = Staff([voice_1, voice_2])
>>> f(staff)
\new Staff {
  \context Voice = "Top Voice" {
    c'4
    d'4
    e'4
    f'4
  }
  \context Voice = "Bottom Voice" {
    \clef "bass"
    c4
    d4
    e4
    f4
  }
}

>>> voice = componenttools.get_component_in_expr_with_name(staff, 'Top Voice')
>>> f(voice)
\context Voice = "Top Voice" {
  c'4
  d'4
  e'4
  f'4
}
```

Return one component.

Raise missing component error when no named component is found.

Raise extra component error when more than one component with *name* is found.

componenttools.get_component_start_offset

`componenttools.get_component_start_offset` (*component*)

New in version 1.1. Get *component* start offset:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> f(staff)
\new Staff {
  c'8
  d'8
  e'8
  f'8
}

>>> componenttools.get_component_start_offset(staff[1])
Offset(1, 8)
```

Return nonnegative fraction.

componenttools.get_component_start_offset_in_seconds

`componenttools.get_component_start_offset_in_seconds` (*component*)

New in version 1.1. Get *component* start offset in seconds:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score = Score([staff])
>>> contexttools.TempoMark(Duration(1, 4), 52)(score)
TempoMark(Duration(1, 4), 52)(Score<<1>>)
>>> f(score)
\new Score <<
  \new Staff {
    \tempo 4=52
    c'8
    d'8
    e'8
    f'8
  }
>>

>>> componenttools.get_component_start_offset_in_seconds(score.leaves[1])
Offset(15, 26)
```

Return nonnegative fraction.

componenttools.get_component_stop_offset

`componenttools.get_component_stop_offset` (*component*)

New in version 1.1. Get *component* stop offset:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> f(staff)
\new Staff {
  c'8
  d'8
  e'8
  f'8
}

>>> componenttools.get_component_stop_offset(staff[1])
Offset(1, 4)
```

Return positive fraction.

componenttools.get_component_stop_offset_in_seconds

`componenttools.get_component_stop_offset_in_seconds` (*component*)

New in version 1.1. Get *component* stop offset in seconds:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score = Score([staff])
>>> contexttools.TempoMark(Duration(1, 4), 52)(score)
TempoMark(Duration(1, 4), 52)(Score<<1>>)
>>> f(score)
\new Score <<
  \new Staff {
```

```

        \tempo 4=52
        c'8
        d'8
        e'8
        f'8
    }
>>

>>> componenttools.get_component_stop_offset_in_seconds(score.leaves[1])
Offset(15, 13)

```

Return positive fraction.

componenttools.get_components_in_expr_with_name

`componenttools.get_components_in_expr_with_name(expr, name)`

New in version 2.9. Get components in *expr* with *name*:

```

>>> staff = Staff(r"\new Voice { c'8 d'8 } \new Voice { e'8 f'8 } \new Voice { g'4 }")
>>> staff[0].name = 'outer voice'
>>> staff[1].name = 'middle voice'
>>> staff[2].name = 'outer voice'

>>> f(staff)
\new Staff {
  \context Voice = "outer voice" {
    c'8
    d'8
  }
  \context Voice = "middle voice" {
    e'8
    f'8
  }
  \context Voice = "outer voice" {
    g'4
  }
}

>>> componenttools.get_components_in_expr_with_name(staff, 'outer voice')
[Voice-"outer voice"{2}, Voice-"outer voice"{1}]

>>> componenttools.get_components_in_expr_with_name(staff, 'middle voice')
[Voice-"middle voice"{2}]

```

Return list of zero or more components found.

componenttools.get_first_component_in_expr_with_name

`componenttools.get_first_component_in_expr_with_name(expr, name)`

New in version 1.1. Get first component in *expr* with *name*:

```

>>> flute_staff = Staff("c'8 d'8 e'8 f'8")
>>> flute_staff.name = 'Flute'
>>> violin_staff = Staff("c'8 d'8 e'8 f'8")
>>> violin_staff.name = 'Violin'

```

```
>>> staff_group = scoretools.StaffGroup([flute_staff, violin_staff])
>>> score = Score([staff_group])

>>> componenttools.get_first_component_in_expr_with_name(score, 'Violin')
Staff-"Violin"{4}
```

Changed in version 2.0: Function returns first component found. Function previously returned tuple of all components found. Changed in version 2.0: renamed `scoretools.find()` to `componenttools.get_first_component_in_expr_with_name()`. Changed in version 2.0: Removed `klass` and `context` keywords. Function operates only on component name.

`componenttools.get_first_component_with_name_in_improper_parentage_of_component`

`componenttools.get_first_component_with_name_in_improper_parentage_of_component` (*component*, *name*)

New in version 2.0. Get first component with *name* in improper parentage of *component*:

```
>>> score = Score([Staff("c'4 d'4 e'4 f'4")])
>>> score.name = 'The Score'

>>> f(score)
\context Score = "The Score" <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
>>

>>> leaf = score.leaves[0]

>>> componenttools.get_first_component_with_name_in_improper_parentage_of_component (
...     leaf, 'The Score')
Score-"The Score"<<1>>

>>> componenttools.get_first_component_with_name_in_improper_parentage_of_component (
...     leaf, 'foo') is None
True
```

Return component or none.

`componenttools.get_first_component_with_name_in_proper_parentage_of_component`

`componenttools.get_first_component_with_name_in_proper_parentage_of_component` (*component*, *name*)

New in version 2.0. Get first component with *name* in proper parentage of *component*:

```
>>> score = Score([Staff("c'4 d'4 e'4 f'4")])
>>> score.name = 'The Score'

>>> f(score)
\context Score = "The Score" <<
  \new Staff {
    c'4
    d'4
```

```

        e'4
        f'4
    }
>>

>>> leaf = score.leaves[0]

>>> componenttools.get_first_component_with_name_in_proper_parentage_of_component (
...     leaf, 'The Score')
Score-"The Score"<<1>>

>>> componenttools.get_first_component_with_name_in_proper_parentage_of_component (
...     leaf, 'foo') is None
True

```

Return component or none.

componenttools.get_first_instance_of_klass_in_improper_parentage_of_component

componenttools.get_first_instance_of_klass_in_improper_parentage_of_component (*component*, *klass*)

New in version 2.0. Get first instance of *klass* in improper parentage of *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> componenttools.get_first_instance_of_klass_in_improper_parentage_of_component (
...     staff[0], Note)
Note("c'8")

```

Return component or none.

componenttools.get_first_instance_of_klass_in_proper_parentage_of_component

componenttools.get_first_instance_of_klass_in_proper_parentage_of_component (*component*, *klass*)

New in version 1.1. Get first instance of *klass* in proper parentage of *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> componenttools.get_first_instance_of_klass_in_proper_parentage_of_component (
...     staff[0], Staff)
Staff{4}

```

Return component or none. Changed in version 2.0: renamed `componenttools.get_first()` to `componenttools.get_first_instance_of_klass_in_proper_parentage_of_component()`.

componenttools.get_improper_contents_of_component

componenttools.get_improper_contents_of_component (*component*)

New in version 2.9. Get improper contents of *component*:

```

>>> staff = Staff("c' d' e' f'")

>>> f(staff)
\new Staff {
    c'4
    d'4

```

```
e'4
f'4
}
```

```
>>> componenttools.get_improper_contents_of_component(staff)
[Staff{4}, Note("c'4"), Note("d'4"), Note("e'4"), Note("f'4")]
```

The functions works for both containers and leaves.

Return a list of *component* together with the proper contents of *component*.

componenttools.get_improper_descendents_of_component

`componenttools.get_improper_descendents_of_component` (*component*)

New in version 2.9. Get improper descendents of *component*:

```
>>> staff = Staff(r"c'4 \times 2/3 { d'8 e'8 f'8 }")

>>> f(staff)
\new Staff {
  c'4
  \times 2/3 {
    d'8
    e'8
    f'8
  }
}

>>> for x in componenttools.get_improper_descendents_of_component(staff):
...     x
...
Staff{2}
Note("c'4")
Tuplet(2/3, [d'8, e'8, f'8])
Note("d'8")
Note("e'8")
Note("f'8")
```

Function returns exactly the same components as `iterationtools.iterate_components_in_expr()`.

Return list of *component* together with proper descendents of *component*.

componenttools.get_improper_descendents_of_component_that_cross_offset

`componenttools.get_improper_descendents_of_component_that_cross_offset` (*component*, *prolated_offset*)

New in version 2.0. Get improper contents of *component* that cross *prolated_offset*:

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
```

```

        d'8
    }
    {
        e'8
        f'8
    }
}

```

Examples refer to the score above.

No components cross prolated offset 0:

```
>>> componenttools.get_improper_descendents_of_component_that_cross_offset(staff, 0)
[]

```

Staff, measure and leaf cross prolated offset 1/16:

```
>>> componenttools.get_improper_descendents_of_component_that_cross_offset(
...     staff, Duration(1, 16))
[Staff{2}, Measure(2/8, [c'8, d'8]), Note("c'8")]

```

Staff and measure cross prolated offset 1/8:

```
>>> componenttools.get_improper_descendents_of_component_that_cross_offset(
...     staff, Duration(1, 8))
[Staff{2}, Measure(2/8, [c'8, d'8])]

```

Staff crosses prolated offset 1/4:

```
>>> componenttools.get_improper_descendents_of_component_that_cross_offset(
...     staff, Duration(1, 4))
[Staff{2}]

```

No components cross prolated offset 99:

```
>>> componenttools.get_improper_descendents_of_component_that_cross_offset(
...     staff, 99)
[]

```

Return list.

componenttools.get_improper_descendents_of_component_that_start_with_component

`componenttools.get_improper_descendents_of_component_that_start_with_component` (*component*)
 New in version 2.9. Get improper contents of *component* that start with *component*:

```
>>> staff = Staff(r"c' << \new Voice { d' } \new Voice { e' } >> f'")

>>> f(staff)
\new Staff {
  c'4
  <<
    \new Voice {
      d'4
    }
    \new Voice {
      e'4
    }
  >>

```



```

        f'4
    }

>>> componenttools.get_improper_descendents_of_component_that_start_with_component (
...     staff[1])
[<<Voice{1}, Voice{1}>>, Voice{1}, Note("d'4"), Voice{1}, Note("e'4")]

```

Return list of *component* together with improper contents that start with *component*.

componenttools.get_improper_descendents_of_component_that_stop_with_component

componenttools.get_improper_descendents_of_component_that_stop_with_component (*component*)

New in version 2.9. Get improper descendents of *component* that stop with *component*:

```

>>> staff = Staff(r"{'c' << \new Voice { d' } \new Voice { e' } >> f'4}")

>>> f(staff)
\new Staff {
  {'c'4
  <<
    \new Voice {
      d'4
    }
    \new Voice {
      e'4
    }
  >>
  f'4
}

>>> componenttools.get_improper_descendents_of_component_that_stop_with_component(staff)
[Staff{3}, Note("f'4")]

```

Return list of *component* together with proper contents that stop with *component*. Changed in version 2.9: re-named `componenttools.get_improper_contents_of_component_that_stop_with_component()` to `componenttools.get_improper_descendents_of_component_that_stop_with_component()`.

componenttools.get_improper_parentage_of_component

componenttools.get_improper_parentage_of_component (*component*)

New in version 1.1. Get improper parentage of *component*:

```

>>> tuplet = Tuplet(Fraction(2, 3), "{'c'8 d'8 e'8}")
>>> staff = Staff([tuplet])
>>> note = staff.leaves[0]

>>> componenttools.get_improper_parentage_of_component(note)
(Note("{'c'8}"), Tuplet(2/3, [{"c'8", "d'8", "e'8"}], Staff{1}))

```

Return tuple of zero or more components.

componenttools.get_improper_parentage_of_component_that_start_with_component

componenttools.get_improper_parentage_of_component_that_start_with_component (*component*)

New in version 2.9. Get improper parentage of *component* that start with *component*:

```
>>> staff = Staff(r"c' << \new Voice { d' } \new Voice { e' } >> f'")

>>> f(staff)
\new Staff {
  c'4
  <<
    \new Voice {
      d'4
    }
    \new Voice {
      e'4
    }
  >>
  f'4
}

>>> componenttools.get_improper_parentage_of_component_that_start_with_component (
...     staff.leaves[1])
[Note("d'4"), Voice{1}, <<Voice{1}, Voice{1}>>]
```

Return list of *component* together with proper parentage that start with *component*.

componenttools.get_improper_parentage_of_component_that_stop_with_component

`componenttools.get_improper_parentage_of_component_that_stop_with_component` (*component*)

New in version 2.9. Get improper parentage of *component* that stop with *component*:

```
>>> staff = Staff(r"c' << \new Voice { d' } \new Voice { e' } >> f'")

f(staff)
\new Staff {
  c'4
  <<
    \new Voice {
      d'4
    }
    \new Voice {
      e'4
    }
  >>
  f'4
}

>>> componenttools.get_improper_parentage_of_component_that_stop_with_component (
...     staff.leaves[-1])
[Note("f'4"), Staff{3}]
```

Return list of *component* with proper parentage that stop with *component*.

componenttools.get_leftmost_components_with_prolated_duration_at_most

`componenttools.get_leftmost_components_with_prolated_duration_at_most` (*components*,

*pro-
lated_duration*)

New in version 2.0. Get leftmost components in *component* with prolated duration at most *prolated_duration*.

Return tuple of `components[:i]` together with the prolated duration of `components[:i]`:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> componenttools.get_leftmost_components_with_prolated_duration_at_most(
...     voice[:], Duration(1, 4))
([Note("c'8"), Note("d'8")], Duration(1, 4))
```

Maximize `i` such that the prolated duration of `components[:i]` is no greater than *prolated_duration*.

Input *components* must be thread-contiguous.

Todo

implement `componenttools.list_leftmost_components_with_prolated_duration_at_least()`.

Todo

implement `componenttools.list_rightmost_components_with_prolated_duration_at_most()`.

Todo

implement `componenttools.list_rightmost_components_with_prolated_duration_at_least()`.

Changed in version 2.0: renamed `componenttools.get_le_duration_prolated()` to `componenttools.get_leftmost_components_with_prolated_duration_at_most()`. Changed in version 2.9: renamed `componenttools.list_leftmost_components_with_prolated_duration_at_most` to `componenttools.get_leftmost_components_with_prolated_duration_at_most()`.

`componenttools.get_likely_multiplier_of_components`

`componenttools.get_likely_multiplier_of_components(components)`

New in version 2.0. Get likely multiplier of *components*:

```
>>> staff = Staff("c'8.. d'8.. e'8.. f'8..")
>>> f(staff)
\new Staff {
    c'8..
    d'8..
    e'8..
    f'8..
}
>>> componenttools.get_likely_multiplier_of_components(staff[:])
Duration(7, 4)
```

Return 1 when no multiplier is likely:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
>>> componenttools.get_likely_multiplier_of_components(staff[:])
Duration(1, 1)
```

Return none when more than one multiplier is likely:

```
>>> staff = Staff(notetools.make_notes([0, 2, 4, 5], [(3, 16), (7, 32)]))
>>> f(staff)
\new Staff {
    c'8.
    d'8..
    e'8.
    f'8..
}
>>> componenttools.get_likely_multiplier_of_components(staff[:]) is None
True
```

Return fraction or none.

componenttools.get_lineage_of_component

`componenttools.get_lineage_of_component` (*component*)

New in version 2.9. Get lineage of *component*:

```
>>> staff = Staff(r"c'4 \times 2/3 { d'8 e'8 f'8 }")

f(staff)
\new Staff {
    c'4
    \times 2/3 {
        d'8
        e'8
        f'8
    }
}

componenttools.get_lineage_of_component(staff[1])
[Staff{2}, Tuplet(2/3, [d'8, e'8, f'8]), Note("d'8"), Note("e'8"), Note("f'8")]
```

Return list of parentage, component and descendents.

componenttools.get_lineage_of_component_that_start_with_component

`componenttools.get_lineage_of_component_that_start_with_component` (*component*)

New in version 2.9. Get lineage of *component* that start with *component*:

```
>>> staff = Staff(r"c' << \new Voice { d'8 e'8 } \new Voice { d''8 e''8 } >> f'4")

>>> f(staff)
\new Staff {
    c'4
    <<
        \new Voice {
            d'8
            e'8
        }
        \new Voice {
            d''8
            e''8
        }
    }
}
```

```

    }
    >>
    f'4
}

>>> staff[1][0]
Voice{2}

>>> componenttools.get_lineage_of_component_that_start_with_component(staff[1][0])
[<<Voice{2}, Voice{2}>>, Voice{2}, Note("d'8")]

```

Return list of all components in the lineage of *component* that start with *component*.

The list always includes *component*.

componenttools.get_lineage_of_component_that_stop_with_component

`componenttools.get_lineage_of_component_that_stop_with_component(component)`

New in version 2.9. Get lineage of *component* that stop with *component*:

```

>>> staff = Staff(r"{'c' << \new Voice { d'8 e'8 } \new Voice { d''8 e''8 } >> f'4}")

>>> f(staff)
\new Staff {
  c'4
  <<
    \new Voice {
      d'8
      e'8
    }
    \new Voice {
      d''8
      e''8
    }
  >>
  f'4
}

>>> staff[1][0]
Voice{2}

>>> componenttools.get_lineage_of_component_that_stop_with_component(staff[1][0])
[<<Voice{2}, Voice{2}>>, Voice{2}, Note("e'8")]

```

Return list of all components in the lineage of *component* that stop with *component*.

The list always includes *component*.

componenttools.get_most_distant_sequential_container_in_improper_parentage_of_component

`componenttools.get_most_distant_sequential_container_in_improper_parentage_of_component(component)`

New in version 2.9. Get first sequential container in the improper parentage of *component* such that the parent of sequential container is either a parallel container or else none:

```

>>> voice_1 = Voice("c'8 d'8")
>>> voice_2 = Voice("e'8 f'8")

```

```
>>> t = Voice([Container([voice_1, voice_2])])
>>> t[0].is_parallel = True
>>> t[0][0].name = 'voice 1'
>>> t[0][1].name = 'voice 2'

>>> f(t)
\new Voice {
  <<
    \context Voice = "voice 1" {
      c'8
      d'8
    }
    \context Voice = "voice 2" {
      e'8
      f'8
    }
  >>
}

>>> note = t.leaves[1]

>>> componenttools.get_most_distant_sequential_container_in_improper_parentage_of_component (
...     note)
Voice-"voice 1"{2}
```

Return none when no such container exists in the improper parentage of *component*.

componenttools.get_nth_component_in_expr

`componenttools.get_nth_component_in_expr (expr, classes, n=0)`

New in version 1.1. Get component *n* in the *classes* of *expr*:

```
>>> staff = Staff([])
>>> durations = [Duration(n, 16) for n in range(1, 5)]
>>> notes = notetools.make_notes([0, 2, 4, 5], durations)
>>> rests = resttools.make_rests(durations)
>>> from abjad.tools import sequencetools
>>> leaves = sequencetools.interlace_sequences(notes, rests)
>>> staff.extend(leaves)

>>> f(staff)
\new Staff {
  c'16
  r16
  d'8
  r8
  e'8.
  r8.
  f'4
  r4
}

>>> for n in range(4):
...     componenttools.get_nth_component_in_expr(staff, Note, n)
...
Note("c'16")
Note("d'8")
```

```

Note("e'8.")
Note("f'4")

>>> for n in range(4):
...     componenttools.get_nth_component_in_expr(staff, Rest, n)
...
Rest('r16')
Rest('r8')
Rest('r8.')
Rest('r4')

>>> componenttools.get_nth_component_in_expr(staff, Staff)
Staff{8}

```

Read right-to-left for negative values of n :

```

>>> for n in range(3, -1, -1):
...     componenttools.get_nth_component_in_expr(staff, Rest, n)
...
Rest('r4')
Rest('r8.')
Rest('r8')
Rest('r16')

```

Return component or none. Changed in version 2.0: renamed `iterate.get_nth()` to `componenttools.get_nth_component_in_expr()`.

`componenttools.get_nth_component_in_time_order_from_component`

`componenttools.get_nth_component_in_time_order_from_component` (*component*, n)

New in version 2.9. Get n th component from *component* in temporal order:

```

>>> staff = Staff(r"c'4 \times 2/3 { d'8 e'8 f'8 } g'2")

>>> f(staff)
\new Staff {
  c'4
  \times 2/3 {
    d'8
    e'8
    f'8
  }
  g'2
}

>>> staff.leaves[1]
Note("d'8")

```

Return component right of *component* for positive n :

```

>>> componenttools.get_nth_component_in_time_order_from_component(
...     staff.leaves[1], 1)
Note("e'8")

>>> componenttools.get_nth_component_in_time_order_from_component(
...     staff.leaves[1], 2)
Note("f'8")

```

```
>>> componenttools.get_nth_component_in_time_order_from_component (
...     staff.leaves[1], 3)
Note("g'2")
```

Return component left of *component* for negative *n*:

```
>>> componenttools.get_nth_component_in_time_order_from_component (
...     staff.leaves[1], -1)
Note("c'4")
```

Return *component* when *n* is 0:

```
>>> componenttools.get_nth_component_in_time_order_from_component (
...     staff.leaves[1], 0)
Note("d'8")
```

Return none when *n* is out of range:

```
>>> componenttools.get_nth_component_in_time_order_from_component (
...     staff.leaves[1], 99) is None
True
```

Return none when *component* has no parent:

```
>>> componenttools.get_nth_component_in_time_order_from_component (
...     staff, 1) is None
True
```

Return component or none.

componenttools.get_nth_namesake_from_component

componenttools.get_nth_namesake_from_component (*component*, *n*)

New in version 2.0. For positive *n*, return namesake to the right of *component*:

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> componenttools.get_nth_namesake_from_component (t[1], 1)
Note("e'8")
```

For negative *n*, return namesake to the left of *component*:

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> componenttools.get_nth_namesake_from_component (t[1], -1)
Note("c'8")
```

Return *component* when *n* is zero:

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> componenttools.get_nth_namesake_from_component (t[1], 0)
Note("d'8")
```

Return component or none.

componenttools.get_nth_sibling_from_component

componenttools.get_nth_sibling_from_component (*component*, *n*)

New in version 2.9. Get *n*th sibling from *component*:


```
>>> staff = Staff("c' d' e' f'")
```

```
>>> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
}
```

```
>>> staff[1]
Note("d'4")
```

Return sibling to the right of *component* for positive *n*:

```
>>> componenttools.get_nth_sibling_from_component(staff[1], 1)
Note("e'4")
```

Return sibling to the left of *component* for negative *n*:

```
>>> componenttools.get_nth_sibling_from_component(staff[1], -1)
Note("c'4")
```

Return *component* when *n* is 0:

```
>>> componenttools.get_nth_sibling_from_component(staff[1], 0)
Note("d'4")
```

Return none when *n* is out of range:

```
>>> componenttools.get_nth_sibling_from_component(staff[1], 99) is None
True
```

Return none when *component* has no parent:

```
>>> componenttools.get_nth_sibling_from_component(staff, 1) is None
True
```

Return component or none.

componenttools.get_parent_and_start_stop_indices_of_components

`componenttools.get_parent_and_start_stop_indices_of_components(components)`

New in version 1.1. Get parent and start / stop indices of *components*:

```
>>> t = Staff("c'8 d'8 e'8 f'8 g'8 a'8")
```

```
>>> f(t)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
}
```

```
>>> leaves = t[-2:]
>>> leaves
[Note("g'8"), Note("a'8")]
>>> componenttools.get_parent_and_start_stop_indices_of_components(leaves)
(Staff{6}, 4, 5)
```

Return parent / start index / stop index triple. Return parent as component or none. Return nonnegative integer start index and nonnegative index stop index. Changed in version 2.0: renamed `componenttools.get_with_indices()` to `componenttools.get_parent_and_start_stop_indices_of_components()`.

componenttools.get_proper_contents_of_component

`componenttools.get_proper_contents_of_component` (*component*)

New in version 2.9. Get proper contents of *component*:

```
>>> staff = Staff("c' d' e' f' ")

>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
}

>>> componenttools.get_proper_contents_of_component(staff)
[Note("c'4"), Note("d'4"), Note("e'4"), Note("f'4")]
```

The function works on leaves:

```
>>> componenttools.get_proper_contents_of_component(staff[0])
[]
```

Return list of the proper contents of component.

componenttools.get_proper_descendents_of_component

`componenttools.get_proper_descendents_of_component` (*component*)

New in version 2.9. Get proper descendents of *component*:

```
>>> staff = Staff(r"c'4 \times 2/3 { d'8 e'8 f'8 }")

>>> f(staff)
\new Staff {
  c'4
  \times 2/3 {
    d'8
    e'8
    f'8
  }
}

>>> componenttools.get_proper_descendents_of_component(staff)
[Note("c'4"), Tuplet(2/3, [d'8, e'8, f'8]), Note("d'8"), Note("e'8"), Note("f'8")]
```

Return list of proper descendents of *component*.

componenttools.get_proper_parentage_of_component

`componenttools.get_proper_parentage_of_component` (*component*)

New in version 1.1. Get proper parentage of *component*:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> staff = Staff([tuplet])
>>> note = staff.leaves[0]
>>> componenttools.get_proper_parentage_of_component(note)
(FixedDurationTuplet(1/4, [c'8, d'8, e'8]), Staff{1})
```

Return tuple of zero or more components.

componenttools.is_immediate_temporal_successor_of_component

`componenttools.is_immediate_temporal_successor_of_component` (*component*, *expr*)

New in version 2.9. True when *expr* is immediate temporal successor of *component*.

Otherwise false.

componenttools.is_orphan_component

`componenttools.is_orphan_component` (*component*)

New in version 1.1. True when *component* has no parent. Otherwise false:

```
>>> note = Note("c'4")
>>> componenttools.is_orphan_component(note)
True
```

Return boolean. Changed in version 2.0: renamed `componenttools.component_is_orphan()` to `componenttools.is_orphan_component()`.

componenttools.is_well_formed_component

`componenttools.is_well_formed_component` (*expr*, *allow_empty_containers=True*)

New in version 1.1. True when *component* is well formed:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)
>>> componenttools.is_well_formed_component(staff)
True
```

Otherwise false:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> staff[1].written_duration = Duration(1, 4)
>>> beamtools.BeamSpanner(staff[:])
BeamSpanner(c'8, d'4, e'8, f'8)
>>> componenttools.is_well_formed_component(staff)
False
```

Beamed quarter notes are not well formed.

Return boolean.

`componenttools.list_badly_formed_components_in_expr`

`componenttools.list_badly_formed_components_in_expr(expr)`

New in version 1.1. List badly formed components in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> staff[1].written_duration = Duration(1, 4)
>>> beamtools.BeamSpanner(staff[:])
BeamSpanner(c'8, d'4, e'8, f'8)
>>> f(staff)
\new Staff {
  c'8 [
  d'4
  e'8
  f'8 ]
}
>>> componenttools.list_badly_formed_components_in_expr(staff)
[Note("d'4")]
```

Beamed quarter notes are not well formed.

Return newly created list of zero or more components.

`componenttools.move_component_subtree_to_right_in_immediate_parent_of_component`

`componenttools.move_component_subtree_to_right_in_immediate_parent_of_component(component)`

New in version 2.0. Move *component* subtree to right in immediate parent of *component*:

```
>>> voice = Voice("c'8 [ d'8 ] e'8 [ f'8 ]")

>>> f(voice)
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> componenttools.move_component_subtree_to_right_in_immediate_parent_of_component(
... voice[1])

>>> f(voice)
\new Voice {
  c'8 [
  e'8 ]
  d'8 [
  f'8 ]
}
```

Return none.

Todo

add `n = 1` keyword to generalize flipped distance.

Todo

make `componenttools.move_component_subtree_to_right_in_immediate_parent_of_component()` work when spanners attach to children of component:

```
>>> voice = Voice(r"\times 2/3 { c'8 [ d'8 e'8 ] \times 2/3 { f'8 ] g'8 a'8 }")

>>> componenttools.move_component_subtree_to_right_in_immediate_parent_of_component (
... voice[0])

>>> f(voice)
\new Voice {
  \times 2/3 {
    f'8 ]
    g'8
    a'8
  }
  \times 2/3 {
    c'8 [
    d'8
    e'8
  }
}

>>> componenttools.is_well_formed_component(voice)
False
```

Preserve spanners. Changed in version 2.0: renamed `componenttools.flip()` to `componenttools.move_component_subtree_to_right_in_immediate_parent_of_component()`.

`componenttools.move_parentage_and_spanners_from_components_to_components`

`componenttools.move_parentage_and_spanners_from_components_to_components` (*donors*,
re-
cip-
i-
ents)

New in version 1.1. Move parentage and spanners from *donors* to *recipients*.

Give everything from donors to recipients. Almost exactly the same as container setitem logic. This helper works with orphan donors. Container setitem logic can not work with orphan donors. Return donors. Changed in version 2.0: renamed `scoretools.bequeath()` to `componenttools.move_parentage_and_spanners_from_components_to_components()`.

`componenttools.number_is_between_start_and_stop_offsets_of_component`

`componenttools.number_is_between_start_and_stop_offsets_of_component` (*timepoint*,
compo-
nent)

New in version 2.0. True when *timepoint* is within the prolated duration of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> leaf = staff.leaves[0]

>>> componenttools.number_is_between_start_and_stop_offsets_of_component (
... Duration(1, 16), leaf)
True
```

```
>>> componenttools.number_is_between_start_and_stop_offsets_of_component (
...     Duration(1, 12), leaf)
True
```

Otherwise false:

```
>>> componenttools.number_is_between_start_and_stop_offsets_of_component (
...     Duration(1, 4), leaf)
False
```

Return boolean.

componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds

`componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds` (*timepoint*, *com-*
po-
nent)

New in version 2.0. True when *timepoint* is within the duration of *component* in seconds:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.TempoMark(Duration(1, 4), 60, target_context=Staff) (staff)
TempoMark(Duration(1, 4), 60) (Staff{4})

>>> leaf = staff.leaves[0]

>>> componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds (
... 0.1, leaf)
True
```

Otherwise false:

```
>>> componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds (
... 0.5, leaf)
False
```

Return boolean.

componenttools.partition_components_by_durations_exactly

`componenttools.partition_components_by_durations_exactly` (*components*, *dura-*
tions, *cyclic=False*,
in_seconds=False, *over-*
hang=False)

New in version 1.1.

componenttools.partition_components_by_durations_ge

`componenttools.partition_components_by_durations_ge` (*components*, *durations*,
cyclic=False, *in_seconds=False*,
overhang=False)

New in version 1.1. Partition *components* by *durations*.

Example 1. Partition *components* cyclically by prolated *durations*. Keep overhang:

```
>>> string = "abj: | 2/8 c'8 d'8 || 2/8 e'8 f'8 || 2/8 g'8 a'8 || 2/8 b'8 c''8 |"
>>> staff = Staff(string)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
  {
    b'8
    c''8
  }
}

>>> parts = componenttools.partition_components_by_durations_ge(
...     staff.leaves, [Duration(3, 16), Duration(1, 16)], cyclic=True, overhang=True)

>>> for part in parts:
...     part
...
[Note("c'8"), Note("d'8")]
[Note("e'8")]
[Note("f'8"), Note("g'8")]
[Note("a'8")]
[Note("b'8"), Note("c''8")]
```

Return list of lists.

Function works not just on components but on any durated objects including spanners.

componenttools.partition_components_by_durations_le

```
componenttools.partition_components_by_durations_le(components, durations,
                                                    cyclic=False, in_seconds=False,
                                                    overhang=False)
```

New in version 1.1.

componenttools.remove_component_subtree_from_score_and_spanners

```
componenttools.remove_component_subtree_from_score_and_spanners(components)
```

New in version 1.1. Example 1. Remove one leaf from score:

```
>>> voice = Voice("c'8 [ { d'8 e'8 } f'8 ]")
>>> spannertools.GlissandoSpanner(voice.leaves)
GlissandoSpanner(c'8, d'8, e'8, f'8)
```

```
>>> f(voice)
\new Voice {
  c'8 [ \glissando
  {
    d'8 \glissando
    e'8 \glissando
  }
  f'8 ]
}

>>> componenttools.remove_component_subtree_from_score_and_spanners(voice.leaves[1:2])
(Note("d'8"),)

>>> f(voice)
\new Voice {
  c'8 [ \glissando
  {
    e'8 \glissando
  }
  f'8 ]
}
```

Example 2. Remove contiguous leaves from score:

```
>>> voice = Voice("c'8 [ { d'8 e'8 } f'8 ]")
>>> spannertools.GlissandoSpanner(voice.leaves)
GlissandoSpanner(c'8, d'8, e'8, f'8)

>>> f(voice)
\new Voice {
  c'8 [ \glissando
  {
    d'8 \glissando
    e'8 \glissando
  }
  f'8 ]
}

>>> componenttools.remove_component_subtree_from_score_and_spanners(voice.leaves[:2])
(Note("c'8"), Note("d'8"))

>>> f(voice)
\new Voice {
  {
    e'8 [ \glissando
  }
  f'8 ]
}
```

Example 3. Remove noncontiguous leaves from score:

```
>>> voice = Voice("c'8 [ { d'8 e'8 } f'8 ]")
>>> spannertools.GlissandoSpanner(voice.leaves)
GlissandoSpanner(c'8, d'8, e'8, f'8)

>>> f(voice)
\new Voice {
  c'8 [ \glissando
  {

```



```

        d'8 \glissando
        e'8 \glissando
    }
    f'8 ]
}

>>> componenttools.remove_component_subtree_from_score_and_spanners(
... [voice.leaves[0], voice.leaves[2]])
[Note("c'8"), Note("e'8")]

>>> f(voice)
\new Voice {
    { d'8 [ \glissando
    }
    f'8 ]
}

```

Example 4. Remove container from score:

```

>>> voice = Voice("c'8 [ { d'8 e'8 } f'8 ]")
>>> spannertools.GlissandoSpanner(voice.leaves)
GlissandoSpanner(c'8, d'8, e'8, f'8)

>>> f(voice)
\new Voice {
    c'8 [ \glissando
    {
        d'8 \glissando
        e'8 \glissando
    }
    f'8 ]
}

>>> componenttools.remove_component_subtree_from_score_and_spanners(voice[1:2])
[{d'8, e'8}]

>>> f(voice)
\new Voice {
    c'8 [ \glissando
    f'8 ]
}

```

Withdraw *components* and children of *components* from spanners.

Return either tuple or list of *components* and children of *components*.

Todo

regularize return value of function.

Changed	in	version	2.0:	renamed	<code>componenttools.detach()</code>	to
					<code>componenttools.remove_component_subtree_from_score_and_spanners()</code> .	

`componenttools.replace_components_with_children_of_components`

`componenttools.replace_components_with_children_of_components` (*components*)
 New in version 1.1. Remove arbitrary *components* from score but retain children of *components* in score:

```
>>> staff = Staff(Container(notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> spannertools.SlurSpanner(staff[:])
SlurSpanner({c'8, d'8}, {e'8, f'8})
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  {
    c'8 [ (
      d'8
    )
  }
  {
    e'8
    f'8 ] )
  }
}

>>> componenttools.replace_components_with_children_of_components(staff[0:1])
[{}]
```

```
>>> f(staff)
\new Staff {
  c'8 [ (
    d'8
    {
      e'8
      f'8 ] )
    }
  }
}
```

Return *components*. Changed in version 2.0: renamed `componenttools.slip()` to `componenttools.replace_components_with_children_of_components()`.

componenttools.report_component_format_contributions

`componenttools.report_component_format_contributions` (*component*, *verbose=False*)

New in version 1.1. Report *component* format contributions as string.

Set *verbose* to True or False.

componenttools.split_component_at_offset

`componenttools.split_component_at_offset` (*component*, *offset*, *fracture_spanners=False*, *tie_split_notes=True*, *tie_split_rests=False*)

New in version 1.1. Split *component* at *offset*.

Example 1. Split *component* at *offset*. Don't fracture spanners:

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> beamtools.BeamSpanner(staff[0])
BeamSpanner(|2/8(2)|)
>>> beamtools.BeamSpanner(staff[1])
```

```

BeamSpanner(|2/8(2)|)
>>> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 [ (
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}

>>> halves = componenttools.split_component_at_offset(
... staff.leaves[0], (1, 32), fracture_spanners=False, tie_split_notes=False)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'32 [ (
    c'16.
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}

```

Example 2. Split component at offset at fracture crossing spanners:

```

>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> beamtools.BeamSpanner(staff[0])
BeamSpanner(|2/8(2)|)
>>> beamtools.BeamSpanner(staff[1])
BeamSpanner(|2/8(2)|)
>>> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 [ (
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}

```

```
>>> halves = componenttools.split_component_at_offset(
... staff.leaves[0], (1, 32), fracture_spanners=True, tie_split_notes=False)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'32 ( ) [
    c'16. (
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}
```

Return pair of left and right part-lists.

componenttools.split_components_at_offsets

`componenttools.split_components_at_offsets` (*components*, *offsets*, *fracture_spanners=False*,
cyclic=False, *tie_split_notes=True*,
tie_split_rests=False)

New in version 2.0. Example 1. Split components cyclically and do not fracture crossing spanners:

```
>>> staff = Staff("abj: | 2/8 c'8 d'8 || 2/8 e'8 f'8 |")

>>> beamtools.BeamSpanner(staff[0])
BeamSpanner(|2/8(2)|)
>>> beamtools.BeamSpanner(staff[1])
BeamSpanner(|2/8(2)|)
>>> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 [ (
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}

>>> componenttools.split_components_at_offsets(
... staff.leaves, [Duration(3, 32)], cyclic=True)
[[Note("c'16."), [Note("c'32"), Note("d'16")]],
 [Note("d'16"), Note("e'32")], [Note("e'16."), [Note("f'16."), [Note("f'32")]]]

>>> f(staff)
\new Staff {
  {
    \time 2/8
```

```

        c'16. [ ( ~
        c'32
        d'16 ~
        d'16 ]
    }
    {
        e'32 [ ~
        e'16.
        f'16. ~
        f'32 ] )
    }
}

```

Example 2. Split components cyclically and fracture spanners:

```

>>> staff = Staff("abj: | 2/8 c'8 d'8 || 2/8 e'8 f'8 |")

>>> beamtools.BeamSpanner(staff[0])
BeamSpanner(|2/8(2)|)
>>> beamtools.BeamSpanner(staff[1])
BeamSpanner(|2/8(2)|)
>>> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 [ (
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}

>>> result = componenttools.split_components_at_offsets(
... staff.leaves, [Duration(3, 32)], cyclic=True, fracture_spanners=True)

>>> result
[[Note("c'16."), [Note("c'32"), Note("d'16")], [Note("d'16"), Note("e'32")],
[Note("e'16."), [Note("f'16."), [Note("f'32")]]]

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'16. ( ) [ ~
    c'32 (
    d'16 ) ~
    d'16 ] (
  }
  {
    e'32 ) [ ~
    e'16. (
    f'16. ) ~
    f'32 ] ( )
  }
}

```

```
    }
}
```

Example 3. Split components once and do not fracture crossing spanners:

```
>>> staff = Staff("abj: | 2/8 c'8 d'8 || 2/8 e'8 f'8 |")

>>> beamtools.BeamSpanner(staff[0])
BeamSpanner(|2/8(2)|)
>>> beamtools.BeamSpanner(staff[1])
BeamSpanner(|2/8(2)|)
>>> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 [ (
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}

>>> offsets = [Duration(1, 32), Duration(3, 32), Duration(5, 32)]

>>> shards = componenttools.split_components_at_offsets(
... staff[:1], offsets, cyclic=False, fracture_spanners=False, tie_split_notes=False)

>>> f(staff)
\new Staff {
  {
    \time 1/32
    c'32 [ (
  }
  {
    \time 3/32
    c'16.
  }
  {
    \time 4/32
    d'8 ]
  }
  {
    \time 2/8
    e'8 [
    f'8 ] )
  }
}
```

Example 4. Split components once and fracture crossing spanners:

```
>>> staff = Staff("abj: | 2/8 c'8 d'8 || 2/8 e'8 f'8 |")
```

```
>>> beamtools.BeamSpanner(staff[0])
BeamSpanner(|2/8(2)|)
>>> beamtools.BeamSpanner(staff[1])
BeamSpanner(|2/8(2)|)
>>> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 [ (
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}
```

```
>>> offsets = [Duration(1, 32), Duration(3, 32), Duration(5, 32)]
>>> shards = componenttools.split_components_at_offsets(
... staff[:1], offsets, cyclic=False, fracture_spanners=True, tie_split_notes=False)

>>> f(staff)
\new Staff {
  {
    \time 1/32
    c'32 [ ] ( )
  }
  {
    \time 3/32
    c'16. [ ] ( )
  }
  {
    \time 4/32
    d'8 [ ] (
  }
  {
    \time 2/8
    e'8 [
    f'8 ] )
  }
}
```

Example 5. Split tupletted components once and fracture crossing spanners:

```
>>> staff = Staff(r"\times 2/3 { c'8 d'8 e'8 } \times 2/3 { f'8 g'8 a'8 }")

>>> beamtools.BeamSpanner(staff[0])
BeamSpanner({c'8, d'8, e'8})
>>> beamtools.BeamSpanner(staff[1])
BeamSpanner({f'8, g'8, a'8})
>>> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8, g'8, a'8)

>>> f(staff)
\new Staff {
```

```

\times 2/3 {
    c'8 [ (
    d'8
    e'8 ]
}
\times 2/3 {
    f'8 [
    g'8
    a'8 ] )
}
}

>>> offsets = [(1, 8)]
>>> shards = componenttools.split_components_at_offsets(
... staff.leaves, offsets, cyclic=False, fracture_spanners=True, tie_split_notes=True)

>>> f(staff)
\new Staff {
    \times 2/3 {
        c'8 [ (
        d'16 ) ~
        d'16 (
        e'8 ]
    }
    \times 2/3 {
        f'8 [
        g'8
        a'8 ] )
    }
}

```

Return list of newly split shards.

Note: Add tests of tupletted notes and rests.

Note: Add examples that show mark and context mark handling.

Note: Add example showing grace and after grace handling.

componenttools.sum_duration_of_components_in_seconds

`componenttools.sum_duration_of_components_in_seconds(components)`

New in version 1.1. Sum duration of *components* in seconds:

```

>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> score = Score([Staff([tuplet])])
>>> contexttools.TempoMark(Duration(1, 4), 48)(score)
TempoMark(Duration(1, 4), 48)(Score<<1>>)
>>> f(score)
\new Score <<
    \new Staff {
        \times 2/3 {
            \tempo 4=48

```



```

        c'8
        d'8
        e'8
    }
}
>>

```

```

>>> componenttools.sum_duration_of_components_in_seconds(tuplet[:])
Duration(5, 4)

```

Changed in version 2.0: renamed `durationtools.sum_seconds()` to `componenttools.sum_duration_of_components_in_seconds()`.

componenttools.sum_preprolated_duration_of_components

`componenttools.sum_preprolated_duration_of_components(components)`

New in version 1.1. Sum preprolated duration of *components*:

```

>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> componenttools.sum_preprolated_duration_of_components(tuplet[:])
Duration(3, 8)

```

Return zero on empty iterable:

```

>>> componenttools.sum_preprolated_duration_of_components([])
0

```

Raise contiguity error on nonparent-contiguous *components*:

```

>>> t = Voice(r"\times 2/3 { c'8 d'8 e'8 } \times 2/3 { f'8 g'8 a'8 }")

>>> f(t)
\new Voice {
  \times 2/3 {
    c'8
    d'8
    e'8
  }
  \times 2/3 {
    f'8
    g'8
    a'8
  }
}
>>> componenttools.sum_preprolated_duration_of_components(t.leaves)
Duration(3, 4)

```

Changed in version 2.0: renamed `componenttools.get_duration_preprolated()` to `componenttools.sum_preprolated_duration_of_components()`.

componenttools.sum_prolated_duration_of_components

`componenttools.sum_prolated_duration_of_components(components)`

New in version 1.1. Sum prolated duration of *components*:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> f(tuplet)
\times 2/3 {
    c'8
    d'8
    e'8
}
>>> componenttools.sum_prolated_duration_of_components(tuplet[:])
Duration(1, 4)
```

Changed in version 2.0: renamed `durationtools.sum_prolated()` to `componenttools.sum_prolated_duration_of_components()`.

`componenttools.tabulate_well_formedness_violations_in_expr`

`componenttools.tabulate_well_formedness_violations_in_expr(expr)`

New in version 1.1. Tabulate well-formedness violations in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> staff[1].written_duration = Duration(1, 4)
>>> beamtools.BeamSpanner(staff[:])
BeamSpanner(c'8, d'4, e'8, f'8)
>>> f(staff)
\new Staff {
    c'8 [
    d'4
    e'8
    f'8 ]
}

>>> componenttools.tabulate_well_formedness_violations_in_expr(staff)
1 /    4 beamed quarter note
0 /    1 discontinuous spanner
0 /    5 duplicate id
0 /    1 empty container
0 /    0 intermarked hairpin
0 /    0 misdurated measure
0 /    0 misfilled measure
0 /    4 mispitched tie
0 /    4 misrepresented flag
0 /    5 missing parent
0 /    0 nested measure
0 /    0 overlapping beam
0 /    0 overlapping glissando
0 /    0 overlapping octavation
0 /    0 short hairpin
```

Beamed quarter notes are not well formed.

`componenttools.yield_components_grouped_by_preprolated_duration`

`componenttools.yield_components_grouped_by_preprolated_duration(components)`

New in version 2.0. Example 1. Yield topmost components grouped by preprolated duration:

```
>>> staff = Staff(r"\times 2/3 { c'4 c'4 c'8 c'16 c'16 } c'16 c'16 c'8 c'8")
```

```

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'4
    c'4
    c'8
    c'16
    c'16
  }
  c'16
  c'16
  c'8
  c'8
}

>>> for x in componenttools.yield_components_grouped_by_preprolated_duration(staff):
...     x
...
(Tuplet(2/3, [c'4, c'4, c'8, c'16, c'16]),)
(Note("c'16"), Note("c'16"))
(Note("c'8"), Note("c'8"))

```

Example 2. Yield topmost components grouped by preprolated duration.

Note that function treats input as a flat sequence and attempts no navigation of the score tree. But it's possible to group components lower in the score tree by passing the output of a component iterator as input to this function:

```

>>> staff = Staff(r"\times 2/3 { c'4 c'4 c'8 c'16 c'16 } c'16 c'16 c'8 c'8")

>>> leaves = iterationtools.iterate_leaves_in_expr(staff)
>>> for x in componenttools.yield_components_grouped_by_preprolated_duration(leaves):
...     x
...
(Note("c'4"), Note("c'4"))
(Note("c'8"),)
(Note("c'16"), Note("c'16"), Note("c'16"), Note("c'16"))
(Note("c'8"), Note("c'8"))

```

Return generator.

Note: Might be best to add `in_sequence` or `topmost` to the name of this function.

componenttools.yield_components_grouped_by_prolated_duration

componenttools.yield_components_grouped_by_prolated_duration (*components*)

New in version 2.0. Example 1. Yield topmost components grouped by prolated duration:

```

>>> staff = Staff(r"\times 2/3 { c'4 c'4 c'8 c'16 c'16 } c'16 c'16 c'8 c'8")

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'4
    c'4
    c'8
    c'16
    c'16
  }

```

```

    }
    c'16
    c'16
    c'8
    c'8
}

>>> for x in componenttools.yield_components_grouped_by_prolated_duration(staff):
...     x
...
(Tuplet(2/3, [c'4, c'4, c'8, c'16, c'16]),)
(Note("c'16"), Note("c'16"))
(Note("c'8"), Note("c'8"))

```

Example 2. Yield topmost components grouped by prolated duration.

Note that function treats input as a flat sequence and attempts no navigation of the score tree. But it's possible to group components lower in the score tree by passing the output of a component iterator as input to this function:

```

>>> staff = Staff(r"\times 2/3 { c'4 c'4 c'8 c'16 c'16 } c'16 c'16 c'8 c'8")

>>> leaves = iterationtools.iterate_leaves_in_expr(staff)
>>> for x in componenttools.yield_components_grouped_by_prolated_duration(leaves):
...     x
...
(Note("c'4"), Note("c'4"))
(Note("c'8"),)
(Note("c'16"), Note("c'16"))
(Note("c'16"), Note("c'16"))
(Note("c'8"), Note("c'8"))

```

Return generator.

Note: Might be best to add `in_sequence` or `topmost` to the name of this function.

componenttools.yield_groups_of_mixed_klasses_in_sequence

`componenttools.yield_groups_of_mixed_klasses_in_sequence` (*sequence*, *klasses*)

New in version 2.0. Example 1. Yield groups of notes and chords at only the top level of score:

```

>>> staff = Staff(r"\times 2/3 { c'8 d'8 r8 } \times 2/3 { r8 <e' g'>8 <f' a'>8 }")
>>> staff.extend("g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8
    d'8
    r8
  }
  \times 2/3 {
    r8
    <e' g'>8
    <f' a'>8
  }
  g'8
  a'8

```

```

    r8
    r8
    <b' d''>8
    <c'' e''>8
}

>>> for group in componenttools.yield_groups_of_mixed_klasses_in_sequence(
...     staff, (Note, Chord)):
...     group
(Note("g'8"), Note("a'8"))
(Chord("<b' d''>8"), Chord("<c'' e''>8"))

```

Example 2. Yield groups of notes and chords at all levels of score:

```

>>> leaves = iterationtools.iterate_leaves_in_expr(staff)

>>> for group in componenttools.yield_groups_of_mixed_klasses_in_sequence(
...     leaves, (Note, Chord)):
...     group
(Note("c'8"), Note("d'8"))
(Chord("<e' g'>8"), Chord("<f' a'>8"), Note("g'8"), Note("a'8"))
(Chord("<b' d''>8"), Chord("<c'' e''>8"))

```

Return generator.

componenttools.yield_topmost_components_grouped_by_type

componenttools.yield_topmost_components_grouped_by_type(*expr*)

New in version 2.0. Example 1. Yield topmost components in *expr* grouped by type:

```

>>> staff = Staff(r"\times 2/3 { c'8 d'8 r8 } \times 2/3 { r8 <e' g'>8 <f' a'>8 }")
>>> staff.extend("g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8
    d'8
    r8
  }
  \times 2/3 {
    r8
    <e' g'>8
    <f' a'>8
  }
  g'8
  a'8
  r8
  r8
  <b' d''>8
  <c'' e''>8
}

>>> for x in componenttools.yield_topmost_components_grouped_by_type(staff):
...     x
(Tuplet(2/3, [c'8, d'8, r8]), Tuplet(2/3, [r8, <e' g'>8, <f' a'>8]))
(Note("g'8"), Note("a'8"))

```

```
(Rest('r8'), Rest('r8'))
(Chord("<b' d''>8"), Chord("<c'' e''>8"))
```

Example 2. Yield leaves at all score levels in *expr* grouped by type:

```
>>> leaves = iterationtools.iterate_leaves_in_expr(staff)

>>> for x in componenttools.yield_topmost_components_grouped_by_type(leaves):
...     x
(Note("c'8"), Note("d'8"))
(Rest('r8'), Rest('r8'))
(Chord("<e' g'>8"), Chord("<f' a'>8"))
(Note("g'8"), Note("a'8"))
(Rest('r8'), Rest('r8'))
(Chord("<b' d''>8"), Chord("<c'' e''>8"))
```

Return generator.

componenttools.yield_topmost_components_of_class_grouped_by_type

`componenttools.yield_topmost_components_of_class_grouped_by_type(expr, klass)`

New in version 2.0. Example 1. Yield runs of topmost notes in *expr*:

```
>>> staff = Staff(r"\times 2/3 { c'8 d'8 r8 } \times 2/3 { r8 <e' g'>8 <f' a'>8 }")
>>> staff.extend("g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8
    d'8
    r8
  }
  \times 2/3 {
    r8
    <e' g'>8
    <f' a'>8
  }
  g'8
  a'8
  r8
  r8
  <b' d''>8
  <c'' e''>8
}

>>> for group in componenttools.yield_topmost_components_of_class_grouped_by_type(
...     staff, Note):
...     group
(Note("g'8"), Note("a'8"))
```

Example 2. Yield runs of notes at all levels in *expr*:

```
>>> leaves = iterationtools.iterate_leaves_in_expr(staff)

>>> for group in componenttools.yield_topmost_components_of_class_grouped_by_type(
...     leaves, Note):
```

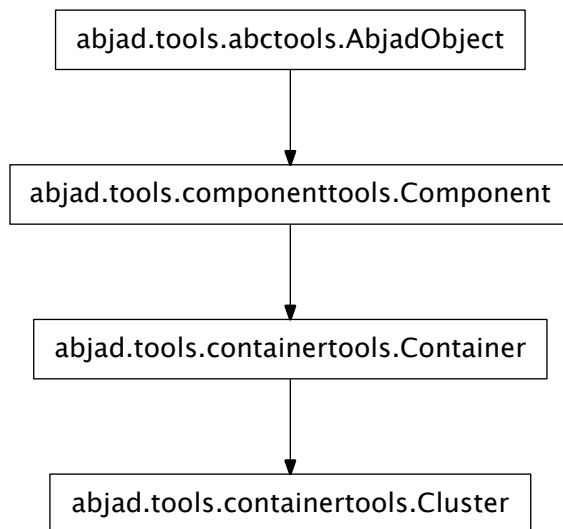
```
...    group
(Note("c'8"), Note("d'8"))
(Note("g'8"), Note("a'8"))
```

Return generator.

containertools

concrete classes

containertools.Cluster



class `containertools.Cluster` (*music=None*, ***kwargs*)
 New in version 1.1. Abjad model of a tone cluster container:

```
>>> cluster = containertools.Cluster("c'8 d'8 b'8")

>>> cluster
Cluster(c'8, d'8, b'8)

>>> f(cluster)
\makeClusters {
  c'8
  d'8
  b'8
}
```

Return cluster object.

Read-only Properties

Cluster.contents_duration

Inherited from `containertools.Container`

Cluster.duration_in_seconds

Inherited from `containertools.Container`

Cluster.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

Cluster.lilypond_format

Cluster.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

Cluster.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Cluster.parent

Inherited from `componenttools.Component`

Cluster.preprolated_duration

Inherited from `containertools.Container`

Cluster.prolated_duration

Inherited from `componenttools.Component`

Cluster.prolation

Inherited from `componenttools.Component`

Cluster.set

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Cluster.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Cluster.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

Cluster.start_offset_in_seconds

Read-only start offset of comonent in seconds.

Inherited from `componenttools.Component`

Cluster.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

Cluster.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Cluster.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Cluster.is_parallel

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])
```

```
>>> f(container)
```

```
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
>>> container.is_parallel
```

```
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True
```

```
>>> f(container)
```

```
<<
```

```
\new Voice {
  c'8
  d'8
  e'8
}
\new Voice {
  g4.
}
>>
```

Return none.

Inherited from `containertools.Container`

Methods

`Cluster.append(component)`

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

`Cluster.extend(expr)`

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: *expr* may now be a LilyPond input string. Inherited from `containertools.Container`

`Cluster.index(component)`

Index *component* in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

`Cluster.insert(i, component)`

Insert *component* in container at index *i*:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
  c'8 [
  cs'8
  d'8
  e'8 ]
}
```

Return none.

Inherited from `containertools.Container`

`Cluster.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
}

>>> container.pop(-1)
Note("e'8")
```

```
>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

`Cluster.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

`Cluster.__add__(expr)`

Concatenate containers *self* and *expr*. The operation `c = a + b` returns a new `Container` *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

`Cluster.__contains__(expr)`

True if *expr* is in container, otherwise False.

Inherited from `containertools.Container`

`Cluster.__delitem__(i)`

Find component(s) at index or slice *i* in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

`Cluster.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Cluster.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Cluster.__getitem__(i)`

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`Cluster.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Cluster.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`Cluster.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

`Cluster.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Cluster.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`Cluster.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Cluster.__mul__(n)`

Inherited from `componenttools.Component`

`Cluster.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Cluster.__radd__(expr)`

Extend container by contents of `expr` to the right.

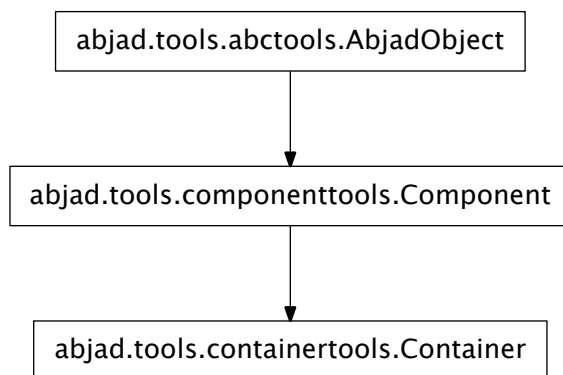
Inherited from `containertools.Container`

`Cluster.__repr__()`

`Cluster.__rmul__(n)`
 Inherited from `componenttools.Component`

`Cluster.__setitem__(i, expr)`
 Set ‘expr’ in self at nonnegative integer index i. Or, set ‘expr’ in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with ‘expr’. Reattach spanners to new contents. This operation leaves all score trees always in tact.
 Inherited from `containertools.Container`

containertools.Container



class `containertools.Container` (*music=None, **kwargs*)

Abjad model of a music container:

```

>>> container = Container("c'8 d'8 e'8 f'8")
>>> f(container)
{
    c'8
    d'8
    e'8
    f'8
}
    
```

Return container object.

Read-only Properties

`Container.contents_duration`

`Container.duration_in_seconds`

`Container.leaves`

Read-only tuple of leaves in container:

```

>>> container = Container("c'8 d'8 e'8")
    
```

```
>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Container.lilypond_format

Inherited from `componenttools.Component`

Container.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Container.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Container.parent

Inherited from `componenttools.Component`

Container.preprolated_duration

Container.prolated_duration

Inherited from `componenttools.Component`

Container.prolation

Inherited from `componenttools.Component`

Container.set

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Container.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Container.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

Container.start_offset_in_seconds

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

Container.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

Container.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`Container.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`Container.is_parallel`

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])
```

```
>>> f(container)
```

```
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
>>> container.is_parallel
```

```
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True
```

```
>>> f(container)
```

```
<<
```

```
\new Voice {
  c'8
  d'8
  e'8
}
\new Voice {
  g4.
}
>>
```

Return none.

Methods

`Container.append(component)`

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
```

```
{
  c'8 [
```



```

        d'8
        e'8 ]
    }

    >>> container.append(Note("f'8"))

    >>> f(container)
    {
        c'8 [
        d'8
        e'8 ]
        f'8
    }

```

Return none.

`Container.extend(expr)`

Extend *expr* against container:

```

    >>> container = Container("c'8 d'8 e'8")
    >>> beam = beamtools.BeamSpanner(container.music)

    >>> f(container)
    {
        c'8 [
        d'8
        e'8 ]
    }

    >>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

    >>> f(container)
    {
        c'8 [
        d'8
        e'8 ]
        cs'8
        ds'8
        es'8
    }

```

Return none. New in version 2.3: *expr* may now be a LilyPond input string.

`Container.index(component)`

Index *component* in container:

```

    >>> container = Container("c'8 d'8 e'8")

    >>> note = container[-1]
    >>> note
    Note("e'8")

    >>> container.index(note)
    2

```

Return nonnegative integer.

`Container.insert(i, component)`

Insert *component* in container at index *i*:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

`Container.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

`Container.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Special Methods

`Container.__add__(expr)`

Concatenate containers self and expr. The operation $c = a + b$ returns a new Container c with the content of both a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

`Container.__contains__(expr)`

True if expr is in container, otherwise False.

`Container.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

`Container.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Container.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Container.__getitem__(i)`

Return component at index i in container. Shallow traversal of container for numeric indices only.

`Container.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Container.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

`Container.__imul__(total)`

Multiply contents of container 'total' times. Return multiplied container.

`Container.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Container.__len__()`

Return nonnegative integer number of components in container.

`Container.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Container.__mul__(n)`

Inherited from `componenttools.Component`

`Container.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Container.__radd__(expr)`

Extend container by contents of `expr` to the right.

`Container.__repr__()`

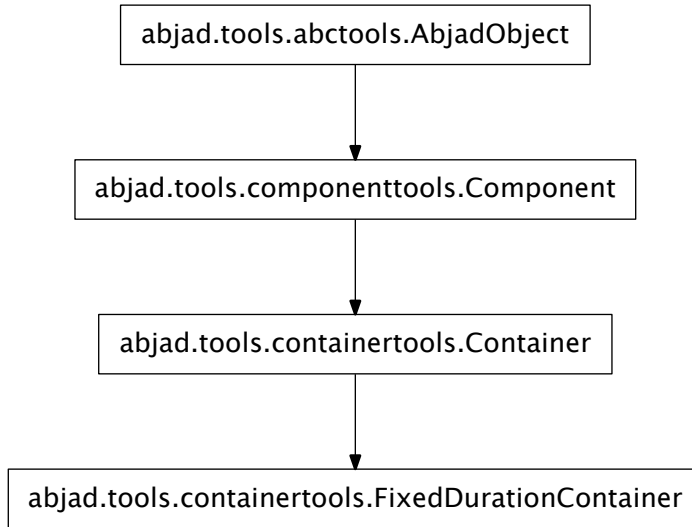
String format of container for interpreter display.

`Container.__rmul__(n)`

Inherited from `componenttools.Component`

`Container.__setitem__(i, expr)`

Set 'expr' in self at nonnegative integer index `i`. Or, set 'expr' in self at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

containertools.FixedDurationContainer

class `containertools.FixedDurationContainer` (*target_duration*, *music=None*, ***kwargs*)

New in version 2.9. Fixed-duration container:

```

>>> container = containertools.FixedDurationContainer((3, 8), "c'8 d'8 e'8")

>>> container
FixedDurationContainer(Duration(3, 8), [Note("c'8"), Note("d'8"), Note("e'8")])

>>> f(container)
{
    c'8
    d'8
    e'8
}
  
```

Fixed-duration containers extend container behavior with format-time checking against a user-specified target duration.

Return fixed-duration container.

Read-only Properties

`FixedDurationContainer.contents_duration`

Inherited from `containertools.Container`

`FixedDurationContainer.duration_in_seconds`

Inherited from `containertools.Container`

`FixedDurationContainer.is_full`

True when preprolated duration equals target duration.

`FixedDurationContainer.is_misfilled`

True when preprolated duration does not equal target duration.

`FixedDurationContainer.is_overfull`

True when preprolated duration is greater than target duration.

`FixedDurationContainer.is_underfull`

True when preprolated duration is less than target duration.

`FixedDurationContainer.leaves`

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

`FixedDurationContainer.lilypond_format`

Read-only LilyPond format of fixed-duration container.

`FixedDurationContainer.music`

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

`FixedDurationContainer.override`

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

`FixedDurationContainer.parent`

Inherited from `componenttools.Component`

`FixedDurationContainer.preprolated_duration`

Inherited from `containertools.Container`

`FixedDurationContainer.prolated_duration`

Inherited from `componenttools.Component`

`FixedDurationContainer.prolation`

Inherited from `componenttools.Component`

`FixedDurationContainer.set`

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

`FixedDurationContainer.spanners`

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`FixedDurationContainer.start_offset`

Read-only start offset of component.

Inherited from `componenttools.Component`

`FixedDurationContainer.start_offset_in_seconds`

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

`FixedDurationContainer.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`FixedDurationContainer.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`FixedDurationContainer.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`FixedDurationContainer.is_parallel`

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')] )
```

```
>>> f(container)
```

```
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
>>> container.is_parallel
```

```
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True
```

```
>>> f(container)
```

```
<<
```

```
\new Voice {
  c'8
  d'8
  e'8
}
\new Voice {
```

```

        g4.
    }
>>

```

Return none.

Inherited from `containertools.Container`

`FixedDurationContainer.target_duration`

Read / write target duration of fixed-duration container.

Methods

`FixedDurationContainer.append(component)`

Append *component* to container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}

```

Return none.

Inherited from `containertools.Container`

`FixedDurationContainer.extend(expr)`

Extend *expr* against container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```



```

        ds'8
        es'8
    }

```

Return `None`. New in version 2.3: `expr` may now be a LilyPond input string. Inherited from `containertools.Container`

`FixedDurationContainer.index(component)`

Index *component* in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> note = container[-1]
```

```
>>> note
```

```
Note("e'8")
```

```
>>> container.index(note)
```

```
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

`FixedDurationContainer.insert(i, component)`

Insert *component* in container at index *i*:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
```

```
{
    c'8 [
    d'8
    e'8 ]
}
```

```
>>> container.insert(1, Note("cs'8"))
```

```
>>> f(container)
```

```
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return `None`.

Inherited from `containertools.Container`

`FixedDurationContainer.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
```

```
{
    c'8 [
    d'8

```

```

        e'8 ]
    }

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}

```

Return component.

Inherited from `containertools.Container`

`FixedDurationContainer.remove(component)`

Remove *component* from container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}

```

Return none.

Inherited from `containertools.Container`

Special Methods

`FixedDurationContainer.__add__(expr)`

Concatenate containers self and *expr*. The operation `c = a + b` returns a new `Container` *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

`FixedDurationContainer.__contains__(expr)`

True if *expr* is in container, otherwise False.

Inherited from `containertools.Container`

`FixedDurationContainer.__delitem__(i)`

Find component(s) at index or slice *i* in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

`FixedDurationContainer.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`FixedDurationContainer.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FixedDurationContainer.__getitem__(i)`

Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`FixedDurationContainer.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`FixedDurationContainer.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`FixedDurationContainer.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

`FixedDurationContainer.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FixedDurationContainer.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`FixedDurationContainer.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FixedDurationContainer.__mul__(n)`

Inherited from `componenttools.Component`

`FixedDurationContainer.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`FixedDurationContainer.__radd__(expr)`

Extend container by contents of *expr* to the right.

Inherited from `containertools.Container`

`FixedDurationContainer.__repr__()`

`FixedDurationContainer.__rmul__(n)`

Inherited from `componenttools.Component`

`FixedDurationContainer.__setitem__(i, expr)`

Set ‘*expr*’ in self at nonnegative integer index *i*. Or, set ‘*expr*’ in self at slice *i*. Find spanners that dominate self[*i*] and children of self[*i*]. Replace contents at self[*i*] with ‘*expr*’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

functions

`containertools.all_are_containers`

`containertools.all_are_containers(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad containers:

```
>>> containers = 3 * Container("c'8 d'8 e'8")
```

```
>>> containertools.all_are_containers(containers)
True
```

True when *expr* is an empty sequence:

```
>>> containertools.all_are_containers([])
True
```

Otherwise false:

```
>>> containertools.all_are_containers('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

`containertools.delete_contents_of_container`

`containertools.delete_contents_of_container(container)`

Delete contents of *container*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

```
>>> containertools.delete_contents_of_container(staff)
[Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
```

```
>>> f(staff)
\new Staff {
}
```

Return *container* contents. Changed in version 2.0: renamed `containertools.contents_delete()` to `containertools.delete_contents_of_container()`.

`containertools.delete_contents_of_container_starting_at_or_after_offset`

`containertools.delete_contents_of_container_starting_at_or_after_offset` (*container*, *prolated_offset*)

New in version 2.0. Delete contents of *container* starting at or after *prolated_offset*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}

>>> containertools.delete_contents_of_container_starting_at_or_after_offset(
...     staff, Duration(1, 8))
Staff{1}

>>> f(staff)
\new Staff {
  c'8 [ ]
}
```

Return *container*. Changed in version 2.0: renamed `containertools.contents_delete_starting_not_before_prolated_offset()` to `containertools.delete_contents_of_container_starting_at_or_after_offset()`.

`containertools.delete_contents_of_container_starting_before_or_at_offset`

`containertools.delete_contents_of_container_starting_before_or_at_offset` (*container*, *prolated_offset*)

New in version 2.0. Delete contents of *container* starting before or at *prolated_offset*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
```

```

        e'8
        f'8 ]
    }

>>> containertools.delete_contents_of_container_starting_before_or_at_offset (
...     staff, Duration(1, 8))
Staff{2}

>>> f(staff)
\new Staff {
    e'8 [
    f'8 ]
}

```

Return *container*. Changed in version 2.0: renamed `containertools.contents_delete_starting_not_after_prolated_offset` to `containertools.delete_contents_of_container_starting_before_or_at_offset()`.

`containertools.delete_contents_of_container_starting_strictly_after_offset`

`containertools.delete_contents_of_container_starting_strictly_after_offset` (*container*, *prolated_offset*)

New in version 2.0. Delete contents of *container* starting strictly after *prolated_offset*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

>>> containertools.delete_contents_of_container_starting_strictly_after_offset (
...     staff, Duration(1, 8))
Staff{2}

>>> f(staff)
\new Staff {
    c'8 [
    d'8 ]
}

```

Return *container*. Changed in version 2.0: renamed `containertools.contents_delete_starting_after_prolated_offset` to `containertools.delete_contents_of_container_starting_strictly_after_offset()`.

`containertools.delete_contents_of_container_starting_strictly_before_offset`

`containertools.delete_contents_of_container_starting_strictly_before_offset` (*container*, *prolated_offset*)

New in version 2.0. Delete contents of *container* contents starting strictly before *prolated_offset*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}

>>> containertools.delete_contents_of_container_starting_strictly_before_offset(
...     staff, Duration(1, 8))
Staff{3}

>>> f(staff)
\new Staff {
  d'8 [
  e'8
  f'8 ]
}

```

Return *container*. Changed in version 2.0: renamed `containertools.contents_delete_starting_before_prola` to `containertools.delete_contents_of_container_starting_strictly_before_offset()`.

`containertools.eject_contents_of_container`

`containertools.eject_contents_of_container` (*container*)

New in version 2.0. Eject contents of *container*:

```

>>> container = Container("c'8 d'8 e'8 f'8")

>>> f(container)
{
  c'8
  d'8
  e'8
  f'8
}

>>> containertools.eject_contents_of_container(container)
[Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]

>>> container
{}

>>> f(container)
{
}

```

Return list of *container* contents.

containertools.fuse_like_named_contiguous_containers_in_expr

`containertools.fuse_like_named_contiguous_containers_in_expr(expr)`

Fuse like-named contiguous containers in *expr*:

```
>>> staff = Staff(r"\new Voice { c'8 d'8 } \new Voice { e'8 f'8 }")
>>> staff[0].name = 'soprano'
>>> staff[1].name = 'soprano'

>>> f(staff)
\new Staff {
  \context Voice = "soprano" {
    c'8
    d'8
  }
  \context Voice = "soprano" {
    e'8
    f'8
  }
}

>>> containertools.fuse_like_named_contiguous_containers_in_expr(staff)
Staff{1}

>>> f(staff)
\new Staff {
  \context Voice = "soprano" {
    c'8
    d'8
    e'8
    f'8
  }
}
```

Return *expr*. Changed in version 2.0: renamed `fuse.containers_by_reference()` to `containertools.fuse_like_named_contiguous_containers_in_expr()`.

containertools.get_element_starting_at_exactly_offset

`containertools.get_element_starting_at_exactly_offset(container, prolated_offset)`

New in version 2.0. Get *container* element starting at exactly *prolated_offset*:

```
>>> voice = Voice("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")

>>> containertools.get_element_starting_at_exactly_offset(voice, Duration(6, 8))
Note("b'8")
```

Raise missing component error when no *container* element starts at exactly *prolated_offset*. Changed in version 2.0: renamed `containertools.get_element_starting_at_prolated_offset()` to `containertools.get_element_starting_at_exactly_offset()`.

containertools.get_first_container_in_improper_parentage_of_component

`containertools.get_first_container_in_improper_parentage_of_component(component)`

New in version 2.0. Get first container in improper parentage of *component*:


```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

>>> containertools.get_first_container_in_improper_parentage_of_component(staff[1])
Staff{4}
```

Return container or none.

containertools.get_first_container_in_proper_parentage_of_component

containertools.get_first_container_in_proper_parentage_of_component (*component*)
 New in version 2.0. Get first container in proper parentage of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

>>> containertools.get_first_container_in_proper_parentage_of_component(staff[1])
Staff{4}
```

Return container or none.

containertools.get_first_element_starting_at_or_after_offset

containertools.get_first_element_starting_at_or_after_offset (*container*, *pro-*
lated_offset)
 New in version 2.0. Get first *container* element starting at or after *prolated_offset*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> containertools.get_first_element_starting_at_or_after_offset(staff, Duration(1, 8))
Note("d'8")
```

Return component.

Return none when no *container* element starts at or after *prolated_offset*. Changed in version 2.0: renamed `containertools.get_leftmost_element_starting_not_before_prolated_offset()` to `containertools.get_first_element_starting_at_or_after_offset()`.

containertools.get_first_element_starting_before_or_at_offset

`containertools.get_first_element_starting_before_or_at_offset` (*container*, *prolated_offset*)

New in version 2.0. Get first *container* element starting before or at *prolated_offset*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> containertools.get_first_element_starting_before_or_at_offset(staff, Duration(1, 8))
Note("d'8")
```

Return component.

Return none when no *container* element starts before or at *prolated_offset*. Changed in version 2.0: renamed `containertools.get_rightmost_element_starting_not_after_prolated_offset()` to `containertools.get_first_element_starting_before_or_at_offset()`.

containertools.get_first_element_starting_strictly_after_offset

`containertools.get_first_element_starting_strictly_after_offset` (*container*, *prolated_offset*)

New in version 2.0. Get first *container* element starting strictly after *prolated_offset*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> containertools.get_first_element_starting_strictly_after_offset(staff, Duration(1, 8))
Note("e'8")
```

Return component.

Return none when no *container* element starts strictly after *prolated_offset*. Changed in version 2.0: renamed `containertools.get_leftmost_element_starting_after_prolated_offset()` to `containertools.get_first_element_starting_strictly_after_offset()`.

containertools.get_first_element_starting_strictly_before_offset

`containertools.get_first_element_starting_strictly_before_offset` (*container*, *prolated_offset*)

New in version 2.0. Get first *container* element starting strictly before *prolated_offset*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> containertools.get_first_element_starting_strictly_before_offset(staff, Duration(1, 8))
Note("c'8")
```

Return component.

Return none when *container* element starts stircly before *prolated_offset*. Changed in version 2.0: renamed `containertools.get_rightmost_element_starting_before_prolated_offset()` to `containertools.get_first_element_starting_strictly_before_offset()`.

containertools.insert_component

`containertools.insert_component` (*container*, *i*, *component*, *fracture_spanners=False*)

New in version 2.10. Insert *component* into *container* at index *i*.

Example 1. Insert *component* into *container* at index *i*. Do not fracture crossing spanners:

```
>>> staff = Staff("c'8 [ d'8 e'8 f'8 ]")

>>> f(staff)
\new Staff {
  c'8 [
    d'8
    e'8
    f'8 ]
}
```

```
>>> containertools.insert_component(staff, 1, Note("cs'8"), fracture_spanners=False)
Staff{5}
```

```
>>> f(staff)
\new Staff {
  c'8 [
    cs'8
    d'8
    e'8
    f'8 ]
}
```

Example 2. Insert *component* into *container* at index *i*. Fracture crossing spanners.

```
>>> staff = Staff("c'8 [ d'8 e'8 f'8 ]")

>>> f(staff)
\new Staff {
  c'8 [
    d'8
    e'8
    f'8 ]
}
```

```
>>> parts = containertools.insert_component(staff, 1, Rest('r8'), fracture_spanners=True)
```

```
>>> f(staff)
\new Staff {
  c'8 [ ]
  r8
  d'8 [
    e'8
    f'8 ]
}
```

Return *container* or list of fractured spanners.

`containertools.move_parentage_children_and_spanners_from_components_to_empty_container`

`containertools.move_parentage_children_and_spanners_from_components_to_empty_container` (*comp*
con
taine

Move parentage, children and spanners from donor *components* to recipient empty *container*:

```
>>> voice = Voice("{ c'8 [ d'8 ] { e'8 f'8 } { g'8 a'8 ] }")
```

```
>>> f(voice)
\new Voice {
  {
    c'8 [
    d'8
  ]
  {
    e'8
    f'8
  ]
  {
    g'8
    a'8 ]
  ]
}
```

```
>>> tuplet = Tuplet(Fraction(3, 4), [])
>>> containertools.move_parentage_children_and_spanners_from_components_to_empty_container(
... voice[:2], tuplet)

>>> f(voice)
\new Voice {
  \fraction \times 3/4 {
    c'8 [
    d'8
    e'8
    f'8
  ]
  {
    g'8
    a'8 ]
  ]
}
```

Return none.

containertools.remove_leafless_containers_in_expr

containertools.remove_leafless_containers_in_expr(*expr*)

Remove empty containers in *expr*:

```
>>> staff = Staff("{ c'8 d'8 } { e'8 f'8 } { g'8 a'8 } { b'8 c''8 }")
>>> beamtools.BeamSpanner(staff[:])
BeamSpanner({c'8, d'8}, {e'8, f'8}, {g'8, a'8}, {b'8, c''8})

>>> containertools.delete_contents_of_container(staff[1])
[Note("e'8"), Note("f'8")]
>>> containertools.delete_contents_of_container(staff[-1])
[Note("b'8"), Note("c''8")]

>>> f(staff)
\new Staff {
  {
    c'8 [
    d'8
  ]
  {

```

```

    }
    {
        g'8
        a'8 ]
    }
    {
    }
}

>>> containertools.remove_leafless_containers_in_expr(staff)

>>> f(staff)
\new Staff {
    {
        c'8 [
        d'8
    }
    {
        g'8
        a'8 ]
    }
}

```

Return `none`. Changed in version 2.0: renamed `containertools.remove_empty()` to `containertools.remove_leafless_containers_in_expr()`.

containertools.repeat_contents_of_container

`containertools.repeat_contents_of_container(container, total=2)`

New in version 1.1. Repeat contents of *container*:

```

>>> staff = Staff("c'8 d'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8)

>>> f(staff)
\new Staff {
    c'8 [
    d'8 ]
}

>>> containertools.repeat_contents_of_container(staff, 3)
Staff{6}

>>> f(staff)
\new Staff {
    c'8 [
    d'8 ]
    c'8 [
    d'8 ]
    c'8 [
    d'8 ]
}

```

Leave *container* unchanged when *total* is 1.

Empty *container* when *total* is 0.

Return *container*. Changed in version 2.0: renamed `containertools.contents_multiply()` to `containertools.repeat_contents_of_container()`.

`containertools.repeat_last_n_elements_of_container`

`containertools.repeat_last_n_elements_of_container(container, n=1, total=2)`

New in version 1.1. Repeat last *n* elements of *container*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

>>> containertools.repeat_last_n_elements_of_container(staff, n=2, total=3)
Staff{8}

>>> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
    e'8 [
    f'8 ]
    e'8 [
    f'8 ]
}
```

Return *container*. Changed in version 2.0: renamed `containertools.extend_cyclic()` to `containertools.repeat_last_n_elements_of_container()`.

`containertools.replace_container_slice_with_rests`

`containertools.replace_container_slice_with_rests(container, start=None, stop=None, big_endian=True)`

New in version 2.10. Replace *container* slice from *start* to *stop* with big-endian rests.

Example 1. Replace all container elements:

```
>>> container = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b' c''8")

>>> container = containertools.replace_container_slice_with_rests(container)

>>> f(container)
\new Staff {
    r1
}
```

Example 2. Replace container elements from 1 forward:

```
>>> container = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b' c''8")

>>> container = containertools.replace_container_slice_with_rests(
...     container, start=1)

>>> f(container)
\new Staff {
    c'8
    r2..
}
```

Example 3. Replace container elements from 1 to 2:

```
>>> container = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b' c''8")

::

>>> container= containertools.replace_container_slice_with_rests(
...     container, start=1, stop=2)

>>> f(container)
\new Staff {
    c'8
    r8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
}
```

Return *container*.

containertools.replace_contents_of_target_container_with_contents_of_source_container

containertools.replace_contents_of_target_container_with_contents_of_source_container (*target_container*, *source_container*)

New in version 2.0. Replace contents of *target_container* with contents of *source_container*:

```
>>> staff = Staff(Tuplet(Fraction(2, 3), "c'8 d'8 e'8") * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(
...     staff)
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, ... [5] ..., c''8, d''8)

>>> f(staff)
\new Staff {
    \times 2/3 {
        c'8 [
        d'8
        e'8
    ]
    \times 2/3 {
        f'8
        g'8
        a'8
    }
}
```

```

    }
    \times 2/3 {
        b'8
        c''8
        d''8 ]
    }
}

>>> container = Container("c'8 d'8 e'8")
>>> spannertools.SlurSpanner(container.leaves)
SlurSpanner(c'8, d'8, e'8)

>>> f(container)
{
    c'8 (
    d'8
    e'8 )
}

>>> containertools.replace_contents_of_target_container_with_contents_of_source_container(
...     staff[1], container)
Tuplet(2/3, [c'8, d'8, e'8])

>>> f(staff)
\new Staff {
    \times 2/3 {
        c'8 [
        d'8
        e'8
    ]
    \times 2/3 {
        c'8 (
        d'8
        e'8 )
    ]
    \times 2/3 {
        b'8
        c''8
        d''8 ]
    }
}

```

Leave *source_container* empty:

```

>>> container
{}

```

Return *target_container*.

containertools.report_container_modifications

`containertools.report_container_modifications(container)`

Report *container* modifications as string:

```

>>> container = Container("c'8 d'8 e'8 f'8")
>>> container.override.note_head.color = 'red'
>>> container.override.note_head.style = 'harmonic'

```



```
>>> f(container)
{
    \override NoteHead #'color = #red
    \override NoteHead #'style = #'harmonic
    c'8
    d'8
    e'8
    f'8
    \revert NoteHead #'color
    \revert NoteHead #'style
}

>>> string = containertools.report_container_modifications(container)

>>> print string
{
    \override NoteHead #'color = #red
    \override NoteHead #'style = #'harmonic
    %% 4 components omitted %%
    \revert NoteHead #'color
    \revert NoteHead #'style
}
```

Return string.

containertools.reverse_contents_of_container

`containertools.reverse_contents_of_container(container)`

New in version 1.1. Reverse contents of *container*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves[:2])
BeamSpanner(c'8, d'8)
>>> spannertools.SlurSpanner(staff.leaves[2:])
SlurSpanner(e'8, f'8)

>>> f(staff)
\new Staff {
    c'8 [
    d'8 ]
    e'8 (
    f'8 )
}

>>> containertools.reverse_contents_of_container(staff)
Staff{4}

>>> f(staff)
\new Staff {
    f'8 (
    e'8 )
    d'8 [
    c'8 ]
}
```

Return *container*. Changed in version 2.0: renamed `containertools.contents_reverse()` to `containertools.reverse_contents_of_container()`.

containertools.scale_contents_of_container

`containertools.scale_contents_of_container(container, multiplier)`

New in version 1.1. Scale contents of *container* by dot *multiplier*:

```
>>> staff = Staff("c'8 d'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8 ]
}
```

```
>>> containertools.scale_contents_of_container(staff, Duration(3, 2))
Staff{2}
```

```
>>> f(staff)
\new Staff {
  c'8. [
  d'8. ]
}
```

Scale contents of *container* by tie *multiplier*:

```
>>> staff = Staff("c'8 d'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8 ]
}
```

```
>>> containertools.scale_contents_of_container(staff, Duration(5, 4))
Staff{4}
```

```
>>> f(staff)
\new Staff {
  c'8 [ ~
  c'32
  d'8 ~
  d'32 ]
}
```

Scale contents of *container* by nonbinary *multiplier*:

```
>>> staff = Staff("c'8 d'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8 ]
}
```

```
>>> containertools.scale_contents_of_container(staff, Duration(4, 3))
Staff{2}

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'4 [
  ]
  \times 2/3 {
    d'4 ]
  }
}
```

Return *container*. Changed in version 2.0: renamed `containertools.contents_scale()` to `containertools.scale_contents_of_container()`.

containertools.set_container_multiplier

`containertools.set_container_multiplier(container, multiplier)`

Set *container multiplier*:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")

>>> f(tuplet)
\times 2/3 {
  c'8
  d'8
  e'8
}

>>> containertools.set_container_multiplier(tuplet, Duration(3, 4))

>>> f(tuplet)
\fraction \times 3/4 {
  c'8
  d'8
  e'8
}
```

Return *none*. Changed in version 2.0: renamed `containertools.multiplier_set()` to `containertools.set_container_multiplier()`.

containertools.split_container_at_index

`containertools.split_container_at_index(component, i, fracture_spanners=False)`

New in version 1.1. General component index split algorithm. Works on leaves, tuplets, measures, contexts and unqualified containers. Keyword controls spanner behavior at split time. Use `containertools.split_container_at_index_and_fracture_crossing_spanners()` to fracture spanners. Use `containertools.split_container_at_index_and_do_not_fracture_crossing_spanners()` to leave spanners unchanged.

Example 1. Split container and do not fracture crossing spanners:

```
>>> voice = Voice(Measure((3, 8), "c'8 c'8 c'8") * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(voice)
>>> beam = beamtools.BeamSpanner(voice[:])
```

```
>>> f(voice)
\new Voice {
  {
    \time 3/8
    c'8 [
    d'8
    e'8
  ]
  {
    f'8
    g'8
    a'8 ]
  }
}

>>> containertools.split_container_at_index(voice[1], 1, fracture_spanners=False)
(Measure(1/8, [f'8]), Measure(2/8, [g'8, a'8]))

>>> f(voice)
\new Voice {
  {
    \time 3/8
    c'8 [
    d'8
    e'8
  ]
  {
    \time 1/8
    f'8
  ]
  {
    \time 2/8
    g'8
    a'8 ]
  }
}
```

Example 2. Split container and fracture crossing spanners:

```
>>> voice = Voice(tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 c'8 c'8") * 2)
>>> tuplet = voice[1]
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(voice)
>>> beam = beamtools.BeamSpanner(voice[:])

>>> f(voice)
\new Voice {
  \times 2/3 {
    c'8 [
    d'8
    e'8
  ]
  \times 2/3 {
    f'8
    g'8
    a'8 ]
  }
}
```

```

>>> left, right = containertools.split_container_at_index(
...     tuple, 1, fracture_spanners=True)

>>> f(voice)
\new Voice {
  \times 2/3 {
    c'8 [
      d'8
      e'8
    ]
  }
  \times 2/3 {
    f'8 ]
  }
  \times 2/3 {
    g'8 [
      a'8 ]
  }
}

```

Leave spanners and leaves untouched.

Resize resizable containers.

Preserve container multiplier.

Preserve meter denominator.

Return split parts.

`containertools.split_container_by_counts`

`containertools.split_container_by_counts` (*components*, *counts*, *fracture_spanners=False*, *cyclic=False*)

New in version 1.1. Partition Python list of zero or more Abjad components. Partition by zero or more positive integers in counts list. Fracture spanners or not according to keyword. Read counts in list cyclically or not according to keyword. Return list of component lists.

Example 1. Split container cyclically by counts and do not fracture crossing spanners:

```

>>> container = Container("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
>>> voice = Voice([container])
>>> beam = beamtools.BeamSpanner(voice)
>>> slur = spannertools.SlurSpanner(container)

>>> f(voice)
\new Voice {
  {
    c'8 [ (
      d'8
      e'8
      f'8
      g'8
      a'8
      b'8
      c''8 ] )
  }
}

```

```
>>> containertools.split_container_by_counts(
...     container, [1, 3], cyclic=True, fracture_spanners=False)
[[{c'8}], [{d'8, e'8, f'8}], [{g'8}], [{a'8, b'8, c''8}]]

>>> f(voice)
\new Voice {
  {
    c'8 [ (
  }
  {
    d'8
    e'8
    f'8
  }
  {
    g'8
  }
  {
    a'8
    b'8
    c''8 ] )
  }
}
```

Example 2. Split container cyclically by counts and fracture crossing spanners:

```
>>> container = Container("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
>>> voice = Voice([container])
>>> beam = beamtools.BeamSpanner(voice)
>>> slur = spannertools.SlurSpanner(container)

>>> f(voice)
\new Voice {
  {
    c'8 [ (
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8 ] )
  }
}

>>> containertools.split_container_by_counts(
...     container, [1, 3], cyclic=True, fracture_spanners=True)
[[{c'8}], [{d'8, e'8, f'8}], [{g'8}], [{a'8, b'8, c''8}]]

>>> f(voice)
\new Voice {
  {
    c'8 ( ) [
  }
  {
    d'8 (
    e'8
    f'8 )
  }
}
```

```

    }
    {
        g'8 ( )
    }
    {
        a'8 (
        b'8
        c''8 ] )
    }
}

```

Example 3. Split container once by counts and do not fracture crossing spanners:

```

>>> container = Container("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
>>> voice = Voice([container])
>>> beam = beamtools.BeamSpanner(voice)
>>> slur = spannertools.SlurSpanner(container)

>>> f(voice)
\new Voice {
  {
    c'8 [ (
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8 ] )
  }
}

>>> containertools.split_container_by_counts(
...     container, [1, 3], cyclic=False, fracture_spanners=False)
[[{c'8}], [{d'8, e'8, f'8}], [{g'8, a'8, b'8, c''8}]]

>>> f(voice)
\new Voice {
  {
    c'8 [ (
  }
  {
    d'8
    e'8
    f'8
  }
  {
    g'8
    a'8
    b'8
    c''8 ] )
  }
}

```

Example 4. Split container once by counts and fracture crossing spanners:

```

>>> container = Container("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
>>> voice = Voice([container])
>>> beam = beamtools.BeamSpanner(voice)

```

```
>>> slur = spannertools.SlurSpanner(container)

>>> f(voice)
\new Voice {
  {
    c'8 [ (
      d'8
      e'8
      f'8
      g'8
      a'8
      b'8
      c''8 ] )
  }
}

>>> containertools.split_container_by_counts(
...     container, [1, 3], cyclic=False, fracture_spanners=True)
[[{c'8}], [{d'8, e'8, f'8}], [{g'8, a'8, b'8, c''8}]]

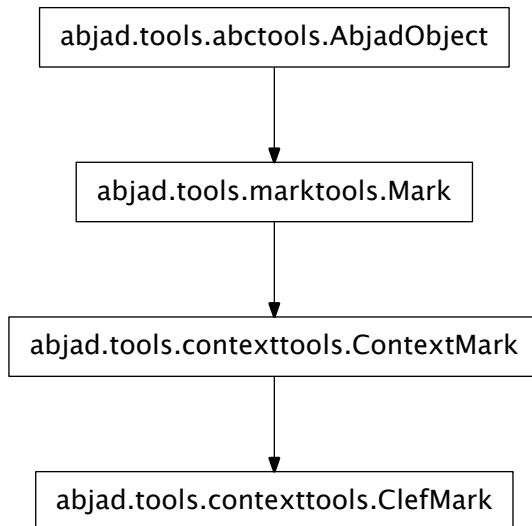
>>> f(voice)
\new Voice {
  {
    c'8 ( ) [
  }
  {
    d'8 (
    e'8
    f'8 )
  }
  {
    g'8 (
    a'8
    b'8
    c''8 ] )
  }
}
```

Return list of split parts.

contexttools

concrete classes

contexttools.ClefMark



class contexttools.**ClefMark** (*arg*, *target_context=None*)

New in version 2.0. Abjad model of a clef:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

>>> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}
  
```

Clef marks target the staff context by default.

Read-only Properties

ClefMark.**effective_context**

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

ClefMark.lilypond_format

Read-only LilyPond format of clef:

```
>>> clef = contexttools.ClefMark('treble')
>>> clef.lilypond_format
'\\clef "treble"'
```

Return string.

ClefMark.middle_c_position

Read-only middle-C position of clef:

```
>>> clef = contexttools.ClefMark('treble')
>>> clef.middle_c_position
-6
```

Return integer number of stafflines.

ClefMark.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

ClefMark.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

ClefMark.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Read/write Properties**ClefMark.clef_name**

Get clef name:

```
>>> clef = contexttools.ClefMark('treble')
>>> clef.clef_name
'treble'
```

Set clef name:

```
>>> clef.clef_name = 'alto'
>>> clef.clef_name
'alto'
```

Return string.

Methods**ClefMark.attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`**ClefMark.detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`**Special Methods****ClefMark.__call__**(**args*)Inherited from `marktools.Mark`**ClefMark.__delattr__**(**args*)Inherited from `marktools.Mark`**ClefMark.__eq__**(*arg*)**ClefMark.__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`**ClefMark.__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

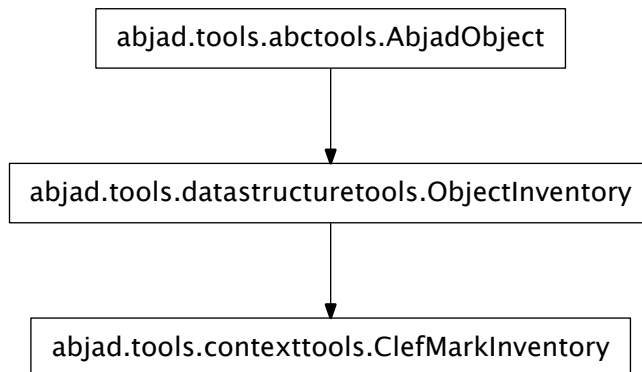
`ClefMark.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`ClefMark.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`ClefMark.__ne__(arg)`
 Inherited from `marktools.Mark`

`ClefMark.__repr__()`
 Inherited from `marktools.Mark`

`contexttools.ClefMarkInventory`



```

class contexttools.ClefMarkInventory(tokens=None, name=None)
  New in version 2.8. Abjad model of an ordered list of clefs:

  >>> inventory = contexttools.ClefMarkInventory(['treble', 'bass'])

  >>> inventory
  ClefMarkInventory([ClefMark('treble'), ClefMark('bass')])

  >>> 'treble' in inventory
  True

  >>> contexttools.ClefMark('treble') in inventory
  True

  >>> 'alto' in inventory
  False
  
```

Clef mark inventories implement list interface and are mutable.

Read-only Properties

`ClefMarkInventory.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`ClefMarkInventory.name`

Read / write name of inventory.

Inherited from `datastructuretools.ObjectInventory`

Methods

`ClefMarkInventory.append(token)`

Change *token* to item and append.

Inherited from `datastructuretools.ObjectInventory`

`ClefMarkInventory.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`ClefMarkInventory.extend(tokens)`

Change *tokens* to items and extend.

Inherited from `datastructuretools.ObjectInventory`

`ClefMarkInventory.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`ClefMarkInventory.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`ClefMarkInventory.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`ClefMarkInventory.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`ClefMarkInventory.reverse()`

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`ClefMarkInventory.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`ClefMarkInventory.__add__()`
`x.__add__(y) <==> x+y`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__contains__(token)`
 Inherited from `datastructuretools.ObjectInventory`

`ClefMarkInventory.__delitem__()`
`x.__delitem__(y) <==> del x[y]`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

`ClefMarkInventory.__eq__()`
`x.__eq__(y) <==> x==y`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__ge__()`
`x.__ge__(y) <==> x>=y`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__getitem__()`
`x.__getitem__(y) <==> x[y]`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

`ClefMarkInventory.__gt__()`
`x.__gt__(y) <==> x>y`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__iadd__()`
`x.__iadd__(y) <==> x+=y`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__imul__()`
`x.__imul__(y) <==> x*=y`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__iter__()` <==> `iter(x)`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__le__()`
`x.__le__(y) <==> x<=y`
 Inherited from `__builtin__.list`

`ClefMarkInventory.__len__()` $\iff \text{len}(x)$
 Inherited from `__builtin__.list`

`ClefMarkInventory.__lt__()`
`x.__lt__(y)` $\iff x < y$
 Inherited from `__builtin__.list`

`ClefMarkInventory.__mul__()`
`x.__mul__(n)` $\iff x * n$
 Inherited from `__builtin__.list`

`ClefMarkInventory.__ne__()`
`x.__ne__(y)` $\iff x \neq y$
 Inherited from `__builtin__.list`

`ClefMarkInventory.__repr__()`
 Inherited from `datastructuretools.ObjectInventory`

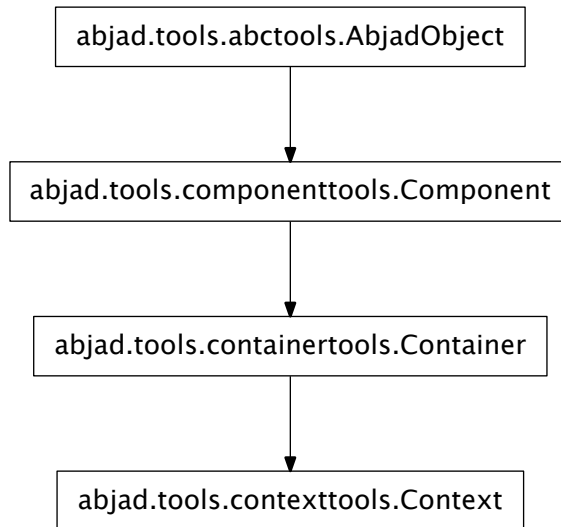
`ClefMarkInventory.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list
 Inherited from `__builtin__.list`

`ClefMarkInventory.__rmul__()`
`x.__rmul__(n)` $\iff n * x$
 Inherited from `__builtin__.list`

`ClefMarkInventory.__setitem__()`
`x.__setitem__(i, y)` $\iff x[i] = y$
 Inherited from `__builtin__.list`

`ClefMarkInventory.__setslice__()`
`x.__setslice__(i, j, y)` $\iff x[i:j] = y$
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

contexttools.Context



class contexttools.**Context** (*music=None, context_name='Context', name=None*)
 New in version 1.0. Abjad model of a horizontal layer of music.

```

>>> context = contexttools.Context(
...     name='MeterVoice', context_name='TimeSignatureContext')

>>> context
TimeSignatureContext-"MeterVoice"{}

>>> f(context)
\context TimeSignatureContext = "MeterVoice" {
}
  
```

Return context object.

Read-only Properties

Context.**contents_duration**

Inherited from containertools.Container

Context.**duration_in_seconds**

Inherited from containertools.Container

Context.**engraver_consists**

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```

>>> staff = Staff([])
>>> staff.engraver_consists.append('Horizontal_bracket_engraver')
>>> f(staff)
\new Staff \with {
  
```



```

        \consists Horizontal_bracket_engraver
    } {
    }

```

Context.**engraver_removals**

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```

>>> staff = Staff([])
>>> staff.engraver_removals.append('Time_signature_engraver')
>>> f(staff)
\new Staff \with {
    \remove Time_signature_engraver
} {
}

```

Context.**is_semantic**

Context.**leaves**

Read-only tuple of leaves in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))

```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

Context.**lilypond_format**

Context.**music**

Read-only tuple of components in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))

```

Return tuple or zero or more components.

Inherited from `containertools.Container`

Context.**override**

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Context.**parent**

Inherited from `componenttools.Component`

Context.**preprolated_duration**

Inherited from `containertools.Container`

Context.**prolated_duration**

Inherited from `componenttools.Component`

Context.**prolation**

Inherited from `componenttools.Component`

`Context.set`

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

`Context.spanners`

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`Context.start_offset`

Read-only start offset of component.

Inherited from `componenttools.Component`

`Context.start_offset_in_seconds`

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

`Context.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`Context.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`Context.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`Context.context_name`

Read / write name of context as a string.

`Context.is_nonsemantic`

Set indicator of nonsemantic voice:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (5, 16), (5, 16)])
>>> voice = Voice(measures)
>>> voice.name = 'HiddenTimeSignatureVoice'

>>> voice.is_nonsemantic = True

>>> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    s1 * 5/16
  }
}
```

```

    }
}

>>> voice.is_nonsemantic
True

```

Get indicator of nonsemantic voice:

```

>>> voice = Voice([])

>>> voice.is_nonsemantic
False

```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Context.**is_parallel**

Get parallel container:

```

>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')]

>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}

>>> container.is_parallel
False

```

Return boolean.

Set parallel container:

```

>>> container.is_parallel = True

>>> f(container)
<<
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
>>

```

Return none.

Inherited from `containertools.Container`

Context.**name**

Read-write name of context. Must be string or none.

Methods

Context.**append**(*component*)

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

Context.**extend**(*expr*)

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: *expr* may now be a LilyPond input string. Inherited from `containertools.Container`

Context.**index** (*component*)

Index *component* in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

Context.**insert** (*i*, *component*)

Insert *component* in container at index *i*:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
  c'8 [
  cs'8
  d'8
  e'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Context.**pop** (*i=-1*)

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
}

>>> container.pop(-1)
Note("e'8")
```

```
>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

Context. **remove** (*component*)

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

Context. **__add__** (*expr*)

Concatenate containers self and *expr*. The operation `c = a + b` returns a new `Container` *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

Context. **__contains__** (*expr*)

True if *expr* is in container, otherwise False.

Inherited from `containertools.Container`

Context. **__delitem__** (*i*)

Find component(s) at index or slice '*i*' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

Context. **__eq__** (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Context.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Context.__getitem__(i)`

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`Context.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Context.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`Context.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

`Context.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Context.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`Context.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Context.__mul__(n)`

Inherited from `componenttools.Component`

`Context.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Context.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`Context.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

`Context.__rmul__(n)`

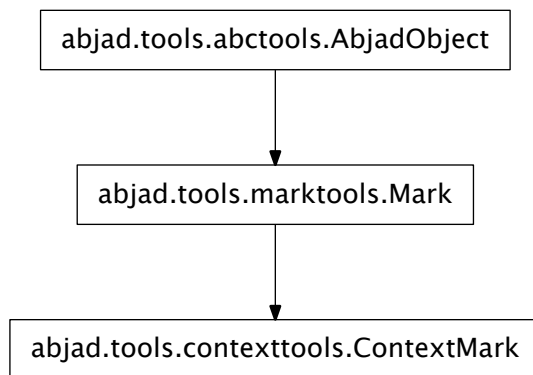
Inherited from `componenttools.Component`

`Context.__setitem__(i, expr)`

Set ‘expr’ in self at nonnegative integer index i. Or, set ‘expr’ in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with ‘expr’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

`contexttools.ContextMark`



class `contexttools.ContextMark` (*target_context=None*)

New in version 2.0. Abstract class from which concrete context marks inherit:

```
>>> note = Note("c'4")
```

```
>>> contexttools.ContextMark()(note)
ContextMark() (c'4)
```

Context marks override `__call__` to attach to Abjad components.

Context marks implement `__slots__`.

Read-only Properties

`ContextMark.effective_context`

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
```

```
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.effective_context is None
True
```


Return context mark or none.

`ContextMark.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`ContextMark.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`ContextMark.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Methods

`ContextMark.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

`ContextMark.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Special Methods

`ContextMark.__call__(*args)`

Inherited from `marktools.Mark`

`ContextMark.__delattr__(*args)`

Inherited from `marktools.Mark`

ContextMark.__eq__(arg)

Inherited from marktools.Mark

ContextMark.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

ContextMark.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

ContextMark.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

ContextMark.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

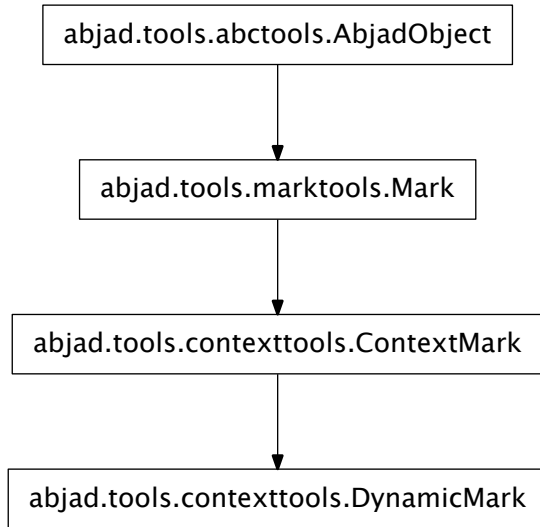
Inherited from abctools.AbjadObject

ContextMark.__ne__(arg)

Inherited from marktools.Mark

ContextMark.__repr__()

Inherited from marktools.Mark

contexttools.DynamicMark

class contexttools.**DynamicMark** (*dynamic_name*, *target_context=None*)
 New in version 2.0. Abjad model of a dynamic mark:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> contexttools.DynamicMark('f')(staff[0])
DynamicMark('f')(c'8)

>>> f(staff)
\new Staff {
  c'8 \f
  d'8
  e'8
  f'8
}
  
```

Dynamic marks target the staff context by default.

Read-only Properties

DynamicMark.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
  
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

DynamicMark.lilypond_format

Read-only LilyPond input format of dynamic mark:

```
>>> dynamic_mark = contexttools.DynamicMark('f')
>>> dynamic_mark.lilypond_format
'\f'
```

Return string.

DynamicMark.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

DynamicMark.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

DynamicMark.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Read/write Properties

DynamicMark.dynamic_name

Get dynamic name:

```
>>> dynamic = contexttools.DynamicMark('f')
>>> dynamic.dynamic_name
'f'
```

Set dynamic name:

```
>>> dynamic.dynamic_name = 'p'
>>> dynamic.dynamic_name
'p'
```

Return string.

Methods

`DynamicMark.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`DynamicMark.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Special Methods

`DynamicMark.__call__(*args)`

`DynamicMark.__delattr__(*args)`

Inherited from `marktools.Mark`

`DynamicMark.__eq__(arg)`

`DynamicMark.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DynamicMark.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DynamicMark.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DynamicMark.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

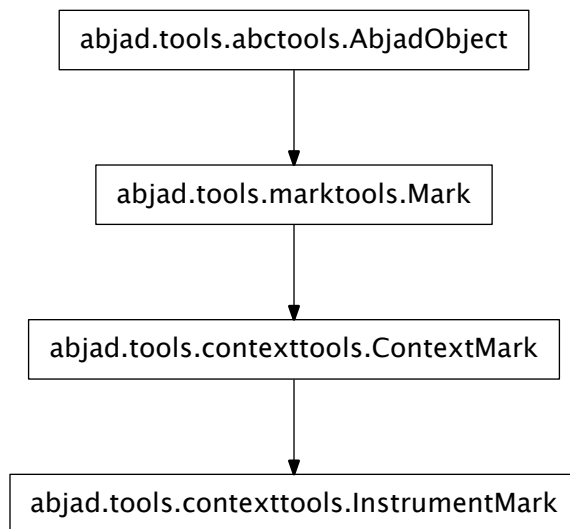
Inherited from `abctools.AbjadObject`

`DynamicMark.__ne__(arg)`

Inherited from `marktools.Mark`

`DynamicMark.__repr__()`
 Inherited from `marktools.Mark`

`contexttools.InstrumentMark`



class `contexttools.InstrumentMark` (*instrument_name*, *short_instrument_name*,
instrument_name_markup=None,
short_instrument_name_markup=None, *target_context=None*)

New in version 2.0. Abjad model of an instrument change:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> contexttools.InstrumentMark('Flute', 'Fl.')(staff)
InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Flute }
  \set Staff.shortInstrumentName = \markup { Fl. }
  c'8
  d'8
  e'8
  f'8
}
    
```

Instrument marks target staff context by default.

Read-only Properties

`InstrumentMark.default_instrument_name`
 Read-only default instrument name.

Return string.

`InstrumentMark.default_short_instrument_name`

Read-only default short instrument name.

Return string.

`InstrumentMark.effective_context`

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`InstrumentMark.lilypond_format`

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

`InstrumentMark.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`InstrumentMark.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`InstrumentMark.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Read/write Properties

`InstrumentMark.instrument_name`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

`InstrumentMark.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

`InstrumentMark.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

`InstrumentMark.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Methods

`InstrumentMark.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`InstrumentMark.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Special Methods

`InstrumentMark.__call__(*args)`

Inherited from `marktools.Mark`

`InstrumentMark.__delattr__(*args)`

Inherited from `marktools.Mark`

`InstrumentMark.__eq__(arg)`

`InstrumentMark.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InstrumentMark.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`InstrumentMark.__hash__()`

`InstrumentMark.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InstrumentMark.__lt__(arg)`

Abjad objects by default do not implement this method.

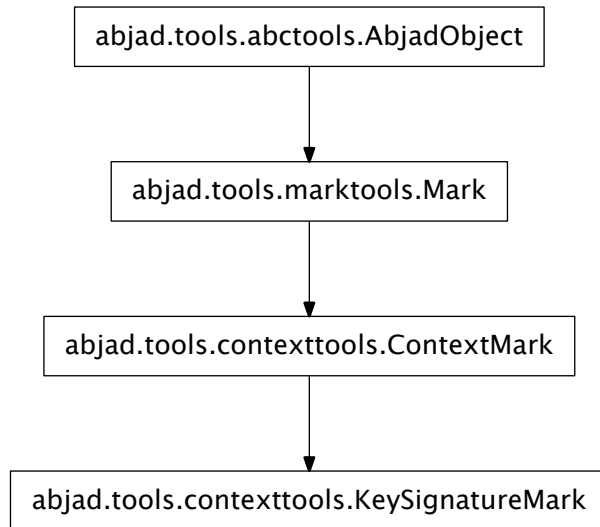
Raise exception.

Inherited from `abctools.AbjadObject`

`InstrumentMark.__ne__(arg)`
 Inherited from `marktools.Mark`

`InstrumentMark.__repr__()`
 Inherited from `marktools.Mark`

`contexttools.KeySignatureMark`



class `contexttools.KeySignatureMark` (*tonic, mode, target_context=None*)
 New in version 2.0. Abjad model of a key signature setting or key signature change:

```

>>> staff = Staff("e'8 fs'8 gs'8 a'8")

>>> contexttools.KeySignatureMark('e', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('e'), Mode('major'))(Staff{4})

>>> f(staff)
\new Staff {
  \key e \major
  e'8
  fs'8
  gs'8
  a'8
}
    
```

Key signature marks target staff context by default.

Read-only Properties

`KeySignatureMark.effective_context`
 Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`KeySignatureMark.lilypond_format`

Read-only LilyPond format of key signature mark:

```
>>> key_signature = contexttools.KeySignatureMark('e', 'major')
>>> key_signature.lilypond_format
'\key e \major'
```

Return string.

`KeySignatureMark.name`

Read-only name of key signature:

```
>>> key_signature = contexttools.KeySignatureMark('e', 'major')
>>> key_signature.name
'E major'
```

Return string.

`KeySignatureMark.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`KeySignatureMark.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`KeySignatureMark.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Read/write Properties

`KeySignatureMark.mode`

Get mode of key signature:

```
>>> key_signature = contexttools.KeySignatureMark('e', 'major')
>>> key_signature.mode
Mode('major')
```

Set mode of key signature:

```
>>> key_signature.mode = 'minor'
>>> key_signature.mode
Mode('minor')
```

Return mode.

`KeySignatureMark.tonic`

Get tonic of key signature:

```
>>> key_signature = contexttools.KeySignatureMark('e', 'major')
>>> key_signature.tonic
NamedChromaticPitchClass('e')
```

Set tonic of key signature:

```
>>> key_signature.tonic = 'd'
>>> key_signature.tonic
NamedChromaticPitchClass('d')
```

Return named chromatic pitch.

Methods

`KeySignatureMark.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`KeySignatureMark.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Special Methods

`KeySignatureMark.__call__(*args)`

Inherited from `marktools.Mark`

KeySignatureMark.__delattr__(*args)
 Inherited from marktools.Mark

KeySignatureMark.__eq__(arg)
 Inherited from abctools.AbjadObject

KeySignatureMark.__ge__(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from abctools.AbjadObject

KeySignatureMark.__gt__(arg)
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from abctools.AbjadObject

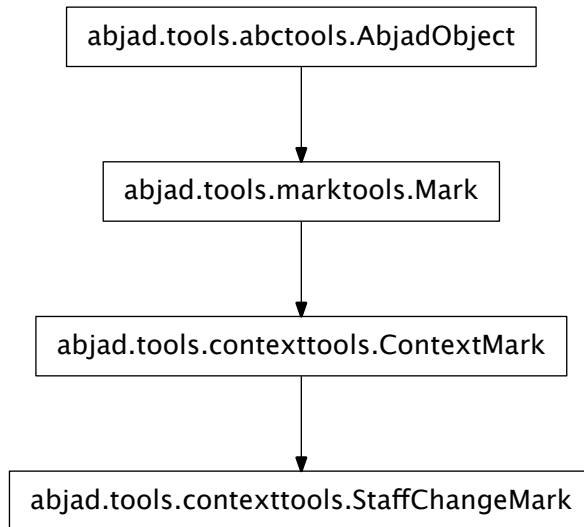
KeySignatureMark.__le__(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from abctools.AbjadObject

KeySignatureMark.__lt__(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from abctools.AbjadObject

KeySignatureMark.__ne__(arg)
 Inherited from marktools.Mark

KeySignatureMark.__repr__()
 Inherited from marktools.Mark

KeySignatureMark.__str__()

contexttools.StaffChangeMark

class contexttools.**StaffChangeMark** (*staff=None, target_context=None*)
New in version 2.0. Abjad model of a staff change:

```
>>> piano_staff = scoretools.PianoStaff([])
>>> rh_staff = Staff("c'8 d'8 e'8 f'8")
>>> rh_staff.name = 'RHStaff'
>>> lh_staff = Staff("s2")
>>> lh_staff.name = 'LHStaff'
>>> piano_staff.extend([rh_staff, lh_staff])

>>> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
    d'8
    e'8
    f'8
  }
  \context Staff = "LHStaff" {
    s2
  }
>>

>>> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

>>> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
```

```

        d'8
        \change Staff = LHStaff
        e'8
        f'8
    }
    \context Staff = "LHStaff" {
        s2
    }
>>

```

Staff change marks target staff context by default.

Read-only Properties

StaffChangeMark.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

StaffChangeMark.lilypond_format

Read-only LilyPond format of staff change mark:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> staff.name = 'RHStaff'
>>> staff_change = contexttools.StaffChangeMark(staff)
>>> staff_change.lilypond_format
'\change Staff = RHStaff'

```

Return string.

StaffChangeMark.start_component

Read-only reference to mark start component:

```

>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")

```

Return component or none.

Inherited from `marktools.Mark`

StaffChangeMark.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

StaffChangeMark.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Read/write Properties

`StaffChangeMark.staff`

Get staff of staff change mark:

```
>>> rh_staff = Staff("c'8 d'8 e'8 f'8")
>>> rh_staff.name = 'RHStaff'
>>> staff_change = contexttools.StaffChangeMark(rh_staff)
>>> staff_change.staff
Staff-"RHStaff"{4}
```

Set staff of staff change mark:

```
>>> lh_staff = Staff("s2")
>>> lh_staff.name = 'LHStaff'
>>> staff_change.staff = lh_staff
>>> staff_change.staff
Staff-"LHStaff"{1}
```

Return staff.

Methods

`StaffChangeMark.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`StaffChangeMark.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Special Methods

`StaffChangeMark.__call__(*args)`

Inherited from `marktools.Mark`

`StaffChangeMark.__delattr__(*args)`
Inherited from `marktools.Mark`

`StaffChangeMark.__eq__(arg)`

`StaffChangeMark.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`StaffChangeMark.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

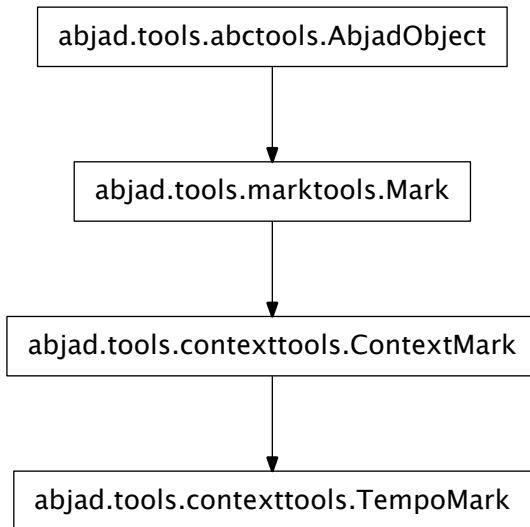
`StaffChangeMark.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`StaffChangeMark.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`StaffChangeMark.__ne__(arg)`
Inherited from `marktools.Mark`

`StaffChangeMark.__repr__()`
Inherited from `marktools.Mark`

contexttools.TempoMark



class contexttools.**TempoMark**(*args, **kwargs)
 New in version 2.0. Abjad model of a tempo indication:

```

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)

>>> contexttools.TempoMark(Duration(1, 8), 52)(staff[0])
TempoMark(Duration(1, 8), 52)(c'8)

>>> f(score)
\new Score <<
  \new Staff {
    \tempo 8=52
    c'8
    d'8
    e'8
    f'8
  }
>>
  
```

Tempo marks target **score** context by default.

Initialization allows many different types of input argument structure.

Read-only Properties

TempoMark.**effective_context**

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

TempoMark.is_imprecise

True if tempo mark is entirely textual, or if tempo mark's `units_per_minute` is a range:

```
>>> contexttools.TempoMark(Duration(1, 4), 60).is_imprecise
False
>>> contexttools.TempoMark('Langsam', 4, 60).is_imprecise
False
>>> contexttools.TempoMark('Langsam').is_imprecise
True
>>> contexttools.TempoMark('Langsam', 4, (35, 50)).is_imprecise
True
>>> contexttools.TempoMark(Duration(1, 4), (35, 50)).is_imprecise
True
```

Return boolean.

TempoMark.lilypond_format

Read-only LilyPond format of tempo mark:

```
>>> tempo = contexttools.TempoMark(Duration(1, 8), 52)
>>> tempo.lilypond_format
'\\tempo 8=52'

>>> tempo.textual_indication = 'Gingerly'
>>> tempo.lilypond_format
'\\tempo Gingerly 8=52'

>>> tempo.units_per_minute = (52, 56)
>>> tempo.lilypond_format
'\\tempo Gingerly 8=52~56'
```

Return string.

TempoMark.quarters_per_minute

Read-only quarters per minute of tempo mark:

```
>>> tempo = contexttools.TempoMark(Duration(1, 8), 52)
>>> tempo.quarters_per_minute
Duration(104, 1)
```

Return fraction, or tuple if `units_per_minute` is a range, or None if tempo mark is imprecise.

TempoMark.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`TempoMark.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`TempoMark.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Read/write Properties

`TempoMark.duration`

Get duration of tempo mark:

```
>>> tempo = contexttools.TempoMark(Duration(1, 8), 52)
>>> tempo.duration
Duration(1, 8)
```

Set duration of tempo mark:

```
>>> tempo.duration = Duration(1, 4)
>>> tempo.duration
Duration(1, 4)
```

Return duration, or None if tempo mark is imprecise.

`TempoMark.textual_indication`

Get textual indication of tempo mark:

```
>>> tempo = contexttools.TempoMark('Langsam', Duration(1, 8), 52)
>>> tempo.textual_indication
'Langsam'
```

Return string or None.

`TempoMark.units_per_minute`

Get units per minute of tempo mark:

```
>>> tempo = contexttools.TempoMark(Duration(1, 8), 52)
>>> tempo.units_per_minute
52
```

Set units per minute of tempo mark:

```
>>> tempo.units_per_minute = 56
>>> tempo.units_per_minute
56
```

Return number.

Methods

`TempoMark.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`TempoMark.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`TempoMark.is_tempo_mark_token(expr)`

True when *expr* has the form of a tempo mark initializer:

```
>>> tempo_mark = contexttools.TempoMark(Duration(1, 4), 72)
>>> tempo_mark.is_tempo_mark_token((Duration(1, 4), 84))
True
```

Otherwise false:

```
>>> tempo_mark.is_tempo_mark_token(84)
False
```

Return boolean.

Special Methods

`TempoMark.__add__(expr)`

`TempoMark.__call__(*args)`

Inherited from `marktools.Mark`

`TempoMark.__delattr__(*args)`

Inherited from `marktools.Mark`

`TempoMark.__div__(expr)`

`TempoMark.__eq__(expr)`

`TempoMark.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TempoMark.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TempoMark.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`TempoMark.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

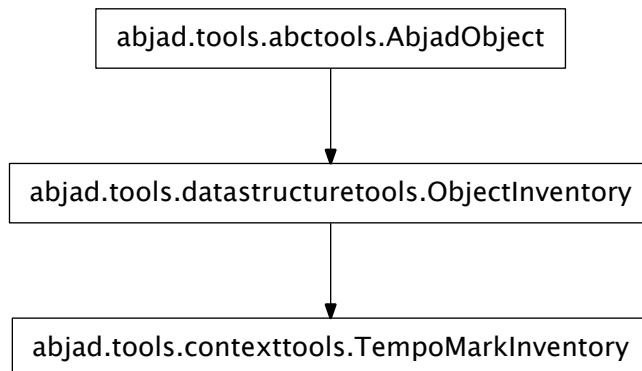
`TempoMark.__mul__(multiplier)`

`TempoMark.__ne__(arg)`
 Inherited from `marktools.Mark`

`TempoMark.__repr__()`
 Inherited from `marktools.Mark`

`TempoMark.__sub__(expr)`

`contexttools.TempoMarkInventory`



class `contexttools.TempoMarkInventory` (*tokens=None, name=None*)

New in version 2.7. Abjad model of an ordered list of tempo marks:

```

>>> inventory = contexttools.TempoMarkInventory([
...     ('Andante', Duration(1, 8), 72),
...     ('Allegro', Duration(1, 8), 84)])

>>> for tempo_mark in inventory:
...     tempo_mark
...
TempoMark('Andante', Duration(1, 8), 72)
TempoMark('Allegro', Duration(1, 8), 84)
  
```

Tempo mark inventories implement list interface and are mutable.

Read-only Properties

`TempoMarkInventory.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`TempoMarkInventory.name`

Read / write name of inventory.

Inherited from `datastructuretools.ObjectInventory`

Methods

`TempoMarkInventory.append(token)`

Change *token* to item and append.

Inherited from `datastructuretools.ObjectInventory`

`TempoMarkInventory.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`TempoMarkInventory.extend(tokens)`

Change *tokens* to items and extend.

Inherited from `datastructuretools.ObjectInventory`

`TempoMarkInventory.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`TempoMarkInventory.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`TempoMarkInventory.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`TempoMarkInventory.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`TempoMarkInventory.reverse()`

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`TempoMarkInventory.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`TempoMarkInventory.__add__()`

`x.__add__(y)` <==> `x+y`

Inherited from `__builtin__.list`

TempoMarkInventory.__contains__(*token*)
Inherited from `datastructuretools.ObjectInventory`

TempoMarkInventory.__delitem__()
`x.__delitem__(y) <==> del x[y]`
Inherited from `__builtin__.list`

TempoMarkInventory.__delslice__()
`x.__delslice__(i, j) <==> del x[i:j]`
Use of negative indices is not supported.
Inherited from `__builtin__.list`

TempoMarkInventory.__eq__()
`x.__eq__(y) <==> x==y`
Inherited from `__builtin__.list`

TempoMarkInventory.__ge__()
`x.__ge__(y) <==> x>=y`
Inherited from `__builtin__.list`

TempoMarkInventory.__getitem__()
`x.__getitem__(y) <==> x[y]`
Inherited from `__builtin__.list`

TempoMarkInventory.__getslice__()
`x.__getslice__(i, j) <==> x[i:j]`
Use of negative indices is not supported.
Inherited from `__builtin__.list`

TempoMarkInventory.__gt__()
`x.__gt__(y) <==> x>y`
Inherited from `__builtin__.list`

TempoMarkInventory.__iadd__()
`x.__iadd__(y) <==> x+=y`
Inherited from `__builtin__.list`

TempoMarkInventory.__imul__()
`x.__imul__(y) <==> x*=y`
Inherited from `__builtin__.list`

TempoMarkInventory.__iter__() <==> *iter(x)*
Inherited from `__builtin__.list`

TempoMarkInventory.__le__()
`x.__le__(y) <==> x<=y`
Inherited from `__builtin__.list`

TempoMarkInventory.__len__() <==> *len(x)*
Inherited from `__builtin__.list`

TempoMarkInventory.__lt__()
`x.__lt__(y) <==> x<y`
Inherited from `__builtin__.list`

TempoMarkInventory.**__mul__**()
 $x._\text{mul}_(n) \iff x*n$
 Inherited from `__builtin__.list`

TempoMarkInventory.**__ne__**()
 $x._\text{ne}_(y) \iff x!=y$
 Inherited from `__builtin__.list`

TempoMarkInventory.**__repr__**()
 Inherited from `datastructuretools.ObjectInventory`

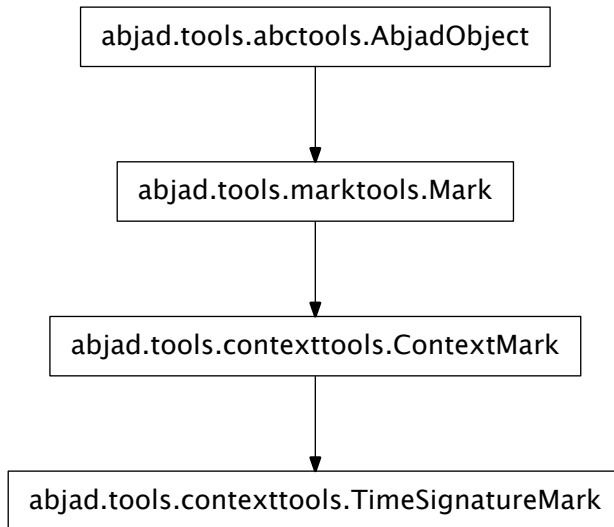
TempoMarkInventory.**__reversed__**()
`L.__reversed__()` – return a reverse iterator over the list
 Inherited from `__builtin__.list`

TempoMarkInventory.**__rmul__**()
 $x._\text{rmul}_(n) \iff n*x$
 Inherited from `__builtin__.list`

TempoMarkInventory.**__setitem__**()
 $x._\text{setitem}_(i, y) \iff x[i]=y$
 Inherited from `__builtin__.list`

TempoMarkInventory.**__setslice__**()
 $x._\text{setslice}_(i, j, y) \iff x[i:j]=y$
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

contexttools.TimeSignatureMark



class contexttools.**TimeSignatureMark**(*args, **kwargs)

New in version 2.0. Abjad model of a time signature:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> contexttools.TimeSignatureMark((4, 8))(staff[0])
TimeSignatureMark((4, 8))(c'8)

>>> f(staff)
\new Staff {
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}
  
```

Abjad time signature marks target **staff context** by default.

Initialize time signature marks to **score context** like this:

```

>>> contexttools.TimeSignatureMark((4, 8), target_context = Score)
TimeSignatureMark((4, 8), target_context = Score)
  
```

Time signatures are immutable.

Read-only Properties

TimeSignatureMark.duration

Read-only duration of time signature mark:

```
>>> time_signature = contexttools.TimeSignatureMark((3, 8))
>>> time_signature.duration
Duration(3, 8)
```

Return fraction.

TimeSignatureMark.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

TimeSignatureMark.is_nonbinary

Read-only indicator true when time signature mark is nonbinary:

```
>>> time_signature = contexttools.TimeSignatureMark((3, 8))
>>> time_signature.is_nonbinary
False
```

Return boolean.

TimeSignatureMark.lilypond_format

Read-only LilyPond format of time signature mark:

```
>>> time_signature = contexttools.TimeSignatureMark((3, 8))
>>> time_signature.lilypond_format
'\time 3/8'
```

Return string.

TimeSignatureMark.multiplier

Read-only multiplier of time signature mark:

```
>>> time_signature = contexttools.TimeSignatureMark((3, 8))
>>> time_signature.multiplier
Fraction(1, 1)
```

Return fraction.

TimeSignatureMark.pair

New in version 2.8. Read-only numerator / denominator pair of time signature:

```
>>> time_signature = contexttools.TimeSignatureMark((3, 8))
>>> time_signature.pair
(3, 8)
```

Return length-2 tuple.

TimeSignatureMark.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`TimeSignatureMark.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`TimeSignatureMark.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Read/write Properties

`TimeSignatureMark.denominator`

Get denominator of time signature mark:

```
>>> time_signature = contexttools.TimeSignatureMark((3, 8))
>>> time_signature
TimeSignatureMark((3, 8))
>>> time_signature.denominator
8
```

Set denominator of time signature mark:

```
>>> time_signature.denominator = 16
>>> time_signature.denominator
16
```

Return integer.

`TimeSignatureMark.numerator`

Get numerator of time signature mark:

```
>>> time_signature = contexttools.TimeSignatureMark((3, 8))
>>> time_signature.numerator
3
```

Set numerator of time signature mark:

```
>>> time_signature.numerator = 4
>>> time_signature.numerator
4
```

Set integer.

`TimeSignatureMark.partial`

Get partial measure pick-up of time signature mark:

```
>>> time_signature = contexttools.TimeSignatureMark(
...     (3, 8), partial=Duration(1, 8))
```

```
>>> time_signature.partial
Duration(1, 8)
```

Set partial measure pick-up of time signature mark:

```
>>> time_signature.partial = Duration(1, 4)
>>> time_signature.partial
Duration(1, 4)
```

Set fraction or none.

Methods

TimeSignatureMark.**attach**(*start_component*)

TimeSignatureMark.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Special Methods

TimeSignatureMark.**__call__**(*args)

Inherited from `marktools.Mark`

TimeSignatureMark.**__delattr__**(*args)

Inherited from `marktools.Mark`

TimeSignatureMark.**__eq__**(arg)

TimeSignatureMark.**__ge__**(arg)

TimeSignatureMark.**__gt__**(arg)

TimeSignatureMark.**__le__**(arg)

TimeSignatureMark.**__lt__**(arg)

TimeSignatureMark.**__ne__**(arg)

TimeSignatureMark.**__nonzero__**()

TimeSignatureMark.**__repr__**()

TimeSignatureMark.**__str__**()

functions

contexttools.all_are_contexts

`contexttools.all_are_contexts` (*expr*)

New in version 2.10. True when *expr* is a sequence of Abjad contexts:

```
>>> contexts = 3 * Voice("c'8 d'8 e'8")

>>> contexttools.all_are_contexts(contexts)
True
```

True when *expr* is an empty sequence:

```
>>> contexttools.all_are_contexts([])
True
```

Otherwise false:

```
>>> contexttools.all_are_contexts('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

contexttools.detach_clef_marks_attached_to_component

`contexttools.detach_clef_marks_attached_to_component` (*component*)

New in version 2.3. Detach clef marks attached to *component*:

```
>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> clef_mark = contexttools.ClefMark('treble')
>>> clef_mark.attach(staff)
ClefMark('treble')(Staff{4})

>>> f(staff)
\new Staff {
  \clef "treble"
  c'4
  d'4
  e'4
  f'4
}

>>> contexttools.detach_clef_marks_attached_to_component(staff)
(ClefMark('treble'),)

>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
}
```

Return tuple of zero or more clef marks.

contexttools.detach_context_marks_attached_to_component

`contexttools.detach_context_marks_attached_to_component` (*component*,
klassses=None)

New in version 2.0. Detach context marks attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> clef_mark = contexttools.ClefMark('treble')(staff)
>>> dynamic_mark = contexttools.DynamicMark('p')(staff[0])
>>> f(staff)
\new Staff {
  \clef "treble"
  c'8 \p
  d'8
  e'8
  f'8
}

>>> contexttools.detach_context_marks_attached_to_component(staff[0])
(DynamicMark('p'),)

>>> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}
```

Return tuple of zero or context marks.

contexttools.detach_dynamic_marks_attached_to_component

`contexttools.detach_dynamic_marks_attached_to_component` (*component*)

New in version 2.3. Detach dynamic marks attached to *component*:

```
>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> dynamic_mark = contexttools.DynamicMark('p')
>>> dynamic_mark.attach(staff[0])
DynamicMark('p')(c'4)

>>> f(staff)
\new Staff {
  c'4 \p
  d'4
  e'4
  f'4
}

>>> contexttools.detach_dynamic_marks_attached_to_component(staff[0])
(DynamicMark('p'),)

>>> f(staff)
\new Staff {
  c'4
  d'4
}
```

```

        e'4
        f'4
    }

```

Return tuple of zero or more dynamic marks.

contexttools.detach_instrument_marks_attached_to_component

`contexttools.detach_instrument_marks_attached_to_component` (*component*)

New in version 2.1. Detach instrument marks attached to *component*:

```

>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> instrument_mark = contexttools.InstrumentMark('Violin ', 'Vn. ')
>>> instrument_mark.attach(staff)
InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. ') (Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Violin }
  \set Staff.shortInstrumentName = \markup { Vn. }
  c'4
  d'4
  e'4
  f'4
}

>>> contexttools.detach_instrument_marks_attached_to_component(staff)
(InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. '),)

>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
}

```

Return tuple of zero or more instrument marks.

contexttools.detach_key_signature_marks_attached_to_component

`contexttools.detach_key_signature_marks_attached_to_component` (*component*)

New in version 2.3. Detach key signature marks attached to *component*:

```

>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> key_signature_mark = contexttools.KeySignatureMark('c', 'major')
>>> key_signature_mark.attach(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major')) (Staff{4})

>>> f(staff)
\new Staff {
  \key c \major
  c'4
  d'4
  e'4
}

```



```

        f'4
    }

>>> contexttools.detach_key_signature_marks_attached_to_component(staff)
(KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major')),)

>>> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
}

```

Return tuple of zero or more key signature marks.

contexttools.detach_staff_change_marks_attached_to_component

`contexttools.detach_staff_change_marks_attached_to_component` (*component*)

New in version 2.3. Detach staff change marks attached to *component*:

```

>>> piano_staff = scoretools.PianoStaff([])
>>> rh_staff = Staff("c'8 d'8 e'8 f'8")
>>> rh_staff.name = 'RHStaff'
>>> lh_staff = Staff("s2")
>>> lh_staff.name = 'LHStaff'
>>> piano_staff.extend([rh_staff, lh_staff])
>>> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

>>> f(piano_staff)
\new PianoStaff <<
    \context Staff = "RHStaff" {
        c'8
        d'8
        \change Staff = LHStaff
        e'8
        f'8
    }
    \context Staff = "LHStaff" {
        s2
    }
>>

>>> contexttools.detach_staff_change_marks_attached_to_component(rh_staff[2])
(StaffChangeMark(Staff-"LHStaff"{1}),)

```

Return tuple of zero or more staff change marks.

contexttools.detach_tempo_marks_attached_to_component

`contexttools.detach_tempo_marks_attached_to_component` (*component*)

New in version 2.3. Detach tempo marks attached to *component*:

```
>>> score = Score([])
>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> score.append(staff)

>>> tempo_mark = contexttools.TempoMark(Duration(1, 8), 52)
>>> tempo_mark.attach(staff)
TempoMark(Duration(1, 8), 52) (Staff{4})

>>> f(score)
\new Score <<
  \new Staff {
    \tempo 8=52
    c'4
    d'4
    e'4
    f'4
  }
>>

>>> contexttools.detach_tempo_marks_attached_to_component(staff)
(TempoMark(Duration(1, 8), 52),)

>>> f(score)
\new Score <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
>>
```

Return tuple of zero or more tempo marks.

contexttools.detach_time_signature_marks_attached_to_component

`contexttools.detach_time_signature_marks_attached_to_component` (*component*)
 New in version 2.0. Detach time signature marks attached to *component*:

```
>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> contexttools.TimeSignatureMark((4, 4))(staff[0])
TimeSignatureMark((4, 4)) (c'4)

>>> f(staff)
\new Staff {
  \time 4/4
  c'4
  d'4
  e'4
  f'4
}

>>> contexttools.detach_time_signature_marks_attached_to_component(staff[0])
(TimeSignatureMark((4, 4)),)

>>> f(staff)
\new Staff {
```

```

    c' 4
    d' 4
    e' 4
    f' 4
}

```

Return tuple of zero or more time signature marks.

contexttools.get_clef_mark_attached_to_component

`contexttools.get_clef_mark_attached_to_component` (*component*)

New in version 2.3. Get clef mark attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

>>> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}

>>> contexttools.get_clef_mark_attached_to_component(staff)
ClefMark('treble')(Staff{4})

```

Return clef mark.

Raise missing mark error when no clef mark attached to *component*.

contexttools.get_clef_marks_attached_to_component

`contexttools.get_clef_marks_attached_to_component` (*component*)

New in version 2.3. Get clef marks attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

>>> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}

>>> contexttools.get_clef_marks_attached_to_component(staff)
(ClefMark('treble')(Staff{4}),)

```

Return tuple of zero or more clef marks.

contexttools.get_context_mark_attached_to_component

`contexttools.get_context_mark_attached_to_component` (*component*, *klasses=None*)

New in version 2.3. Get context mark attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

>>> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}

>>> contexttools.get_context_mark_attached_to_component(staff)
ClefMark('treble')(Staff{4})
```

Return context mark.

Raise missing mark error when no context mark attaches to *component*.

contexttools.get_context_marks_attached_to_any_improper_parent_of_component

`contexttools.get_context_marks_attached_to_any_improper_parent_of_component` (*component*)

New in version 2.0. Get all context marks attached to any improper parent of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})
>>> contexttools.DynamicMark('f')(staff[0])
DynamicMark('f')(c'8)

>>> f(staff)
\new Staff {
  \clef "treble"
  c'8 \f
  d'8
  e'8
  f'8
}

>>> contexttools.get_context_marks_attached_to_any_improper_parent_of_component(staff[0])
(DynamicMark('f')(c'8), ClefMark('treble')(Staff{4}))
```

Return tuple.

contexttools.get_context_marks_attached_to_component

`contexttools.get_context_marks_attached_to_component` (*component*, *klasses=None*)

New in version 2.0. Get context marks attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})
>>> contexttools.DynamicMark('p')(staff[0])
DynamicMark('p')(c'8)

>>> f(staff)
\new Staff {
  \clef "treble"
  c'8 \p
  d'8
  e'8
  f'8
}

>>> contexttools.get_context_marks_attached_to_component(staff[0])
(DynamicMark('p')(c'8),)

```

Return tuple of zero or more context marks.

contexttools.get_dynamic_mark_attached_to_component

`contexttools.get_dynamic_mark_attached_to_component` (*component*)

New in version 2.3. Get dynamic mark attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.DynamicMark('p')(staff[0])
DynamicMark('p')(c'8)

>>> f(staff)
\new Staff {
  c'8 \p
  d'8
  e'8
  f'8
}

>>> contexttools.get_dynamic_mark_attached_to_component(staff[0])
DynamicMark('p')(c'8)

```

Return dynamic mark.

Raise missing mark error when no dynamic mark attaches to *component*.

Raise extra mark error when more than one dynamic mark attaches to *component*.

contexttools.get_dynamic_marks_attached_to_component

`contexttools.get_dynamic_marks_attached_to_component` (*component*)

New in version 2.0. Get dynamic marks attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.DynamicMark('p')(staff[0])
DynamicMark('p')(c'8)

```

```
>>> f(staff)
\new Staff {
  c'8 \p
  d'8
  e'8
  f'8
}

>>> contexttools.get_dynamic_marks_attached_to_component(staff[0])
(DynamicMark('p')(c'8),)
```

Return tuple of zero or more dynamic marks.

contexttools.get_effective_clef

`contexttools.get_effective_clef`(*component*)

New in version 2.0. Get effective clef of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

>>> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}

>>> for note in staff:
...     print note, contexttools.get_effective_clef(note)
...
c'8 ClefMark('treble')(Staff{4})
d'8 ClefMark('treble')(Staff{4})
e'8 ClefMark('treble')(Staff{4})
f'8 ClefMark('treble')(Staff{4})
```

Return clef mark or none.

contexttools.get_effective_context_mark

`contexttools.get_effective_context_mark`(*component*, *klass*)

New in version 2.0. Get effective context mark of *klass* from *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.TimeSignatureMark((4, 8))(staff)
TimeSignatureMark((4, 8))(Staff{4})

>>> f(staff)
\new Staff {
  \time 4/8
  c'8
  d'8
  e'8
```

```

    f'8
}

>>> contexttools.get_effective_context_mark(staff[0], contexttools.TimeSignatureMark)
TimeSignatureMark((4, 8))(Staff{4})

```

Return context mark or none.

contexttools.get_effective_dynamic

`contexttools.get_effective_dynamic(component)`

New in version 2.0. Get effective dynamic of *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.DynamicMark('f')(staff[0])
DynamicMark('f')(c'8)

>>> f(staff)
\new Staff {
  c'8 \f
  d'8
  e'8
  f'8
}

>>> for note in staff:
...     print note, contexttools.get_effective_dynamic(note)
...
c'8 DynamicMark('f')(c'8)
d'8 DynamicMark('f')(c'8)
e'8 DynamicMark('f')(c'8)
f'8 DynamicMark('f')(c'8)

```

Return dynamic mark or none.

contexttools.get_effective_instrument

`contexttools.get_effective_instrument(component)`

New in version 2.0. Get effective instrument of *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.InstrumentMark('Flute', 'Fl.')(staff)
InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Flute }
  \set Staff.shortInstrumentName = \markup { Fl. }
  c'8
  d'8
  e'8
  f'8
}

```

```
>>> for note in staff:
...     print note, contexttools.get_effective_instrument(note)
...
c'8 InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.') (Staff{4})
d'8 InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.') (Staff{4})
e'8 InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.') (Staff{4})
f'8 InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.') (Staff{4})
```

Return instrument mark or none.

contexttools.get_effective_key_signature

contexttools.get_effective_key_signature(*component*)

New in version 2.0. Get effective key signature of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.KeySignatureMark('c', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major')) (Staff{4})

>>> f(staff)
\new Staff {
  \key c \major
  c'8
  d'8
  e'8
  f'8
}

>>> for note in staff:
...     note, contexttools.get_effective_key_signature(note)
...
(Note("c'8"), KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major')) (Staff{4}))
(Note("d'8"), KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major')) (Staff{4}))
(Note("e'8"), KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major')) (Staff{4}))
(Note("f'8"), KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major')) (Staff{4}))
```

Return key signature mark or none.

contexttools.get_effective_staff

contexttools.get_effective_staff(*component*)

New in version 2.0. Get effective staff of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> staff.name = 'First Staff'

>>> f(staff)
\context Staff = "First Staff" {
  c'8
  d'8
  e'8
  f'8
}

>>> for note in staff:
...     print note, contexttools.get_effective_staff(note)
```



```
...
c'8 Staff-"First Staff"{4}
d'8 Staff-"First Staff"{4}
e'8 Staff-"First Staff"{4}
f'8 Staff-"First Staff"{4}
```

Return staff or none.

contexttools.get_effective_tempo

contexttools.get_effective_tempo(*component*)

New in version 2.0. Get effective tempo of *component*:

```
>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)
>>> contexttools.TempoMark(Duration(1, 8), 52)(staff[0])
TempoMark(Duration(1, 8), 52)(c'8)

>>> f(score)
\new Score <<
  \new Staff {
    \tempo 8=52
    c'8
    d'8
    e'8
    f'8
  }
>>

>>> for note in staff:
...     print note, contexttools.get_effective_tempo(note)
...
c'8 TempoMark(Duration(1, 8), 52)(c'8)
d'8 TempoMark(Duration(1, 8), 52)(c'8)
e'8 TempoMark(Duration(1, 8), 52)(c'8)
f'8 TempoMark(Duration(1, 8), 52)(c'8)
```

Return tempo mark or none.

contexttools.get_effective_time_signature

contexttools.get_effective_time_signature(*component*)

New in version 2.0. Get effective time signature of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.TimeSignatureMark((4, 8))(staff)
TimeSignatureMark((4, 8))(Staff{4})

>>> f(staff)
\new Staff {
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}
```

```
>>> for note in staff:
...     note, contexttools.get_effective_time_signature(note)
...
(Note("c'8"), TimeSignatureMark((4, 8))(Staff{4}))
(Note("d'8"), TimeSignatureMark((4, 8))(Staff{4}))
(Note("e'8"), TimeSignatureMark((4, 8))(Staff{4}))
(Note("f'8"), TimeSignatureMark((4, 8))(Staff{4}))
```

Return time signature mark or none.

contexttools.get_instrument_mark_attached_to_component

contexttools.get_instrument_mark_attached_to_component(*component*)

New in version 2.1. Get instrument mark attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> violin = contexttools.InstrumentMark('Violin ', 'Vn. ')
>>> violin.attach(staff)
InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. ') (Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Violin }
  \set Staff.shortInstrumentName = \markup { Vn. }
  c'8
  d'8
  e'8
  f'8
}

>>> contexttools.get_instrument_mark_attached_to_component(staff)
InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. ') (Staff{4})
```

Return instrument mark.

Raise missing mark error when no instrument mark attaches to *component*.

contexttools.get_instrument_marks_attached_to_component

contexttools.get_instrument_marks_attached_to_component(*component*)

New in version 2.3. Get instrument marks attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.InstrumentMark('Flute', 'Fl.')(staff)
InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Flute }
  \set Staff.shortInstrumentName = \markup { Fl. }
  c'8
  d'8
  e'8
  f'8
}
```

```
>>> contexttools.get_instrument_marks_attached_to_component(staff)
(InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4}),)
```

Return tuple of zero or more instrument marks.

contexttools.get_key_signature_mark_attached_to_component

`contexttools.get_key_signature_mark_attached_to_component(component)`

New in version 2.3. Get key signature mark attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.KeySignatureMark('c', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4})

>>> f(staff)
\new Staff {
  \key c \major
  c'8
  d'8
  e'8
  f'8
}
```

```
>>> contexttools.get_key_signature_mark_attached_to_component(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4})
```

Return key signature mark.

Raise missing mark error when no key signature mark attaches to component.

contexttools.get_key_signature_marks_attached_to_component

`contexttools.get_key_signature_marks_attached_to_component(component)`

New in version 2.3. Get key signature marks attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.KeySignatureMark('c', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4})

>>> f(staff)
\new Staff {
  \key c \major
  c'8
  d'8
  e'8
  f'8
}

>>> contexttools.get_key_signature_marks_attached_to_component(staff)
(KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4}),)
```

Return tuple of zero or more key signature marks.

contexttools.get_staff_change_mark_attached_to_component

contexttools.get_staff_change_mark_attached_to_component(*component*)

New in version 2.3. Get staff change mark attached to *component*:

```
>>> piano_staff = scoretools.PianoStaff([])
>>> rh_staff = Staff("c'8 d'8 e'8 f'8")
>>> rh_staff.name = 'RHStaff'
>>> lh_staff = Staff("s2")
>>> lh_staff.name = 'LHStaff'
>>> piano_staff.extend([rh_staff, lh_staff])
>>> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

>>> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
    d'8
    \change Staff = LHStaff
    e'8
    f'8
  }
  \context Staff = "LHStaff" {
    s2
  }
>>

>>> contexttools.get_staff_change_mark_attached_to_component(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)
```

Return staff change mark.

Raise missing mark error when no staff change mark attaches to *component*.

contexttools.get_staff_change_marks_attached_to_component

contexttools.get_staff_change_marks_attached_to_component(*component*)

New in version 2.3. Get staff change marks attached to *component*:

```
>>> piano_staff = scoretools.PianoStaff([])
>>> rh_staff = Staff("c'8 d'8 e'8 f'8")
>>> rh_staff.name = 'RHStaff'
>>> lh_staff = Staff("s2")
>>> lh_staff.name = 'LHStaff'
>>> piano_staff.extend([rh_staff, lh_staff])
>>> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

>>> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
    d'8
    \change Staff = LHStaff
    e'8
    f'8
  }
```

```

    }
    \context Staff = "LHStaff" {
        s2
    }
>>

>>> contexttools.get_staff_change_marks_attached_to_component(rh_staff[2])
(StaffChangeMark(Staff-"LHStaff"{1})(e'8),)

```

Return tuple of zero or more staff change marks.

contexttools.get_tempo_mark_attached_to_component

`contexttools.get_tempo_mark_attached_to_component` (*component*)
 New in version 2.3. Get tempo mark attached to *component*:

```

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)

>>> contexttools.TempoMark(Duration(1, 8), 52)(staff)
TempoMark(Duration(1, 8), 52)(Staff{4})

>>> f(score)
\new Score <<
  \new Staff {
    \tempo 8=52
    c'8
    d'8
    e'8
    f'8
  }
>>

>>> contexttools.get_tempo_mark_attached_to_component(staff)
TempoMark(Duration(1, 8), 52)(Staff{4})

```

Return tempo mark.

Raise missing mark error when no tempo mark attaches to *component*.

contexttools.get_tempo_marks_attached_to_component

`contexttools.get_tempo_marks_attached_to_component` (*component*)
 New in version 2.3. Get tempo marks attached to *component*:

```

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)

>>> contexttools.TempoMark(Duration(1, 8), 52)(staff)
TempoMark(Duration(1, 8), 52)(Staff{4})

>>> f(score)
\new Score <<
  \new Staff {

```

```

        \tempo 8=52
        c'8
        d'8
        e'8
        f'8
    }
>>

>>> contexttools.get_tempo_marks_attached_to_component(staff)
(TempoMark(Duration(1, 8), 52)(Staff{4}),)

```

Return tuple of zero or more tempo marks.

contexttools.get_time_signature_mark_attached_to_component

`contexttools.get_time_signature_mark_attached_to_component(component)`
 New in version 2.0. Get time signature mark attached to *component*:

```

>>> measure = Measure((4, 8), "c'8 d'8 e'8 f'8")

>>> f(measure)
{
    \time 4/8
    c'8
    d'8
    e'8
    f'8
}

>>> contexttools.get_time_signature_mark_attached_to_component(measure)
TimeSignatureMark((4, 8))(|4/8, c'8, d'8, e'8, f'8|)

```

Return time signature mark.

Raise missing mark error when no time signature mark attaches to *component*.

contexttools.get_time_signature_marks_attached_to_component

`contexttools.get_time_signature_marks_attached_to_component(component)`
 New in version 2.3. Get time signature marks attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.TimeSignatureMark((2, 4))(staff)
TimeSignatureMark((2, 4))(Staff{4})

>>> f(staff)
\new Staff {
    \time 2/4
    c'8
    d'8
    e'8
    f'8
}

>>> contexttools.get_time_signature_marks_attached_to_component(staff)
(TimeSignatureMark((2, 4))(Staff{4}),)

```

Return tuple of zero or more `time_signature` marks.

`contexttools.is_component_with_clef_mark_attached`

`contexttools.is_component_with_clef_mark_attached(expr)`

New in version 2.3. True when *expr* is a component with clef mark attached:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

>>> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}

>>> contexttools.is_component_with_clef_mark_attached(staff)
True
```

False otherwise:

```
>>> contexttools.is_component_with_clef_mark_attached(staff[0])
False
```

Return boolean.

`contexttools.is_component_with_context_mark_attached`

`contexttools.is_component_with_context_mark_attached(expr, classes=None)`

New in version 2.0. True when *expr* is a component with context mark of *classes* attached:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.TimeSignatureMark((4, 8))(staff[0])
TimeSignatureMark((4, 8))(c'8)
>>> f(staff)
\new Staff {
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}

>>> contexttools.is_component_with_context_mark_attached(staff[0])
True
```

Otherwise false:

```
>>> contexttools.is_component_with_context_mark_attached(staff)
False
```

Return boolean.

contexttools.is_component_with_dynamic_mark_attached**contexttools.is_component_with_dynamic_mark_attached**(*expr*)New in version 2.3. True when *expr* is a component and has a dynamic mark attached:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.DynamicMark('p')(staff[0])
DynamicMark('p')(c'8)

>>> f(staff)
\new Staff {
  c'8 \p
  d'8
  e'8
  f'8
}

>>> contexttools.is_component_with_dynamic_mark_attached(staff[0])
True
```

Otherwise false:

```
>>> contexttools.is_component_with_dynamic_mark_attached(staff)
False
```

Return boolean.

contexttools.is_component_with_instrument_mark_attached**contexttools.is_component_with_instrument_mark_attached**(*expr*)New in version 2.3. True when *expr* is a component with instrument mark attached:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> violin = contexttools.InstrumentMark('Violin ', 'Vn. ')
>>> violin.attach(staff)
InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. ')(Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Violin }
  \set Staff.shortInstrumentName = \markup { Vn. }
  c'8
  d'8
  e'8
  f'8
}

>>> contexttools.is_component_with_instrument_mark_attached(staff)
True
```

Otherwise false:

```
>>> contexttools.is_component_with_instrument_mark_attached(staff[0])
False
```

Return boolean.

contexttools.is_component_with_key_signature_mark_attached

`contexttools.is_component_with_key_signature_mark_attached(expr)`

New in version 2.3. True when *expr* is a component with key signature mark attached:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.KeySignatureMark('c', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4})

>>> f(staff)
\new Staff {
  \key c \major
  c'8
  d'8
  e'8
  f'8
}

>>> contexttools.is_component_with_key_signature_mark_attached(staff)
True
```

Otherwise false:

```
>>> contexttools.is_component_with_key_signature_mark_attached(staff[0])
False
```

Return boolean.

contexttools.is_component_with_staff_change_mark_attached

`contexttools.is_component_with_staff_change_mark_attached(expr)`

New in version 2.3. True when *expr* is a component with staff change mark attached:

```
>>> piano_staff = scoretools.PianoStaff([])
>>> rh_staff = Staff("c'8 d'8 e'8 f'8")
>>> rh_staff.name = 'RHStaff'
>>> lh_staff = Staff("s2")
>>> lh_staff.name = 'LHStaff'
>>> piano_staff.extend([rh_staff, lh_staff])
>>> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

>>> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
    d'8
    \change Staff = LHStaff
    e'8
    f'8
  }
  \context Staff = "LHStaff" {
    s2
  }
>>
```

```
>>> contexttools.is_component_with_staff_change_mark_attached(rh_staff[2])
True
```

Otherwise false:

```
>>> contexttools.is_component_with_staff_change_mark_attached(rh_staff)
False
```

Return boolean.

contexttools.is_component_with_tempo_mark_attached

`contexttools.is_component_with_tempo_mark_attached(expr)`

New in version 2.3. True when *expr* is a component with tempo mark attached:

```
>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)

>>> contexttools.TempoMark(Duration(1, 8), 52)(staff)
TempoMark(Duration(1, 8), 52)(Staff{4})

>>> f(score)
\new Score <<
  \new Staff {
    \tempo 8=52
    c'8
    d'8
    e'8
    f'8
  }
>>

>>> contexttools.is_component_with_tempo_mark_attached(staff)
True
```

Otherwise false:

```
>>> contexttools.is_component_with_tempo_mark_attached(staff[0])
False
```

Return boolean.

contexttools.is_component_with_time_signature_mark_attached

`contexttools.is_component_with_time_signature_mark_attached(expr)`

New in version 2.0. True when *expr* is a component with time signature mark attached:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.TimeSignatureMark((4, 8))(staff[0])
TimeSignatureMark((4, 8))(c'8)

>>> f(staff)
\new Staff {
  \time 4/8
  c'8
  d'8
```

```

    e'8
    f'8
}

```

```

>>> contexttools.is_component_with_time_signature_mark_attached(staff[0])
True

```

Otherwise false:

```

>>> contexttools.is_component_with_time_signature_mark_attached(staff)
False

```

Return boolean.

contexttools.list_clef_names

contexttools.**list_clef_names**()

New in version 2.8. List clef names:

```

>>> contexttools.list_clef_names()
['alto', 'baritone', 'bass', 'mezzosoprano', 'percussion', 'soprano', 'treble']

```

Return list of strings.

contexttools.set_accidental_style_on_sequential_contexts_in_expr

contexttools.**set_accidental_style_on_sequential_contexts_in_expr**(*expr*, *accidental_style*)

New in version 2.0. Set *accidental_style* for sequential semantic contexts in *expr*:

```

>>> score = Score(Staff("c'8 d'8") * 2)
>>> contexttools.set_accidental_style_on_sequential_contexts_in_expr(score, 'forget')

>>> f(score)
\new Score <<
  \new Staff {
    # (set-accidental-style 'forget)
    c'8
    d'8
  }
  \new Staff {
    # (set-accidental-style 'forget)
    c'8
    d'8
  }
>>

```

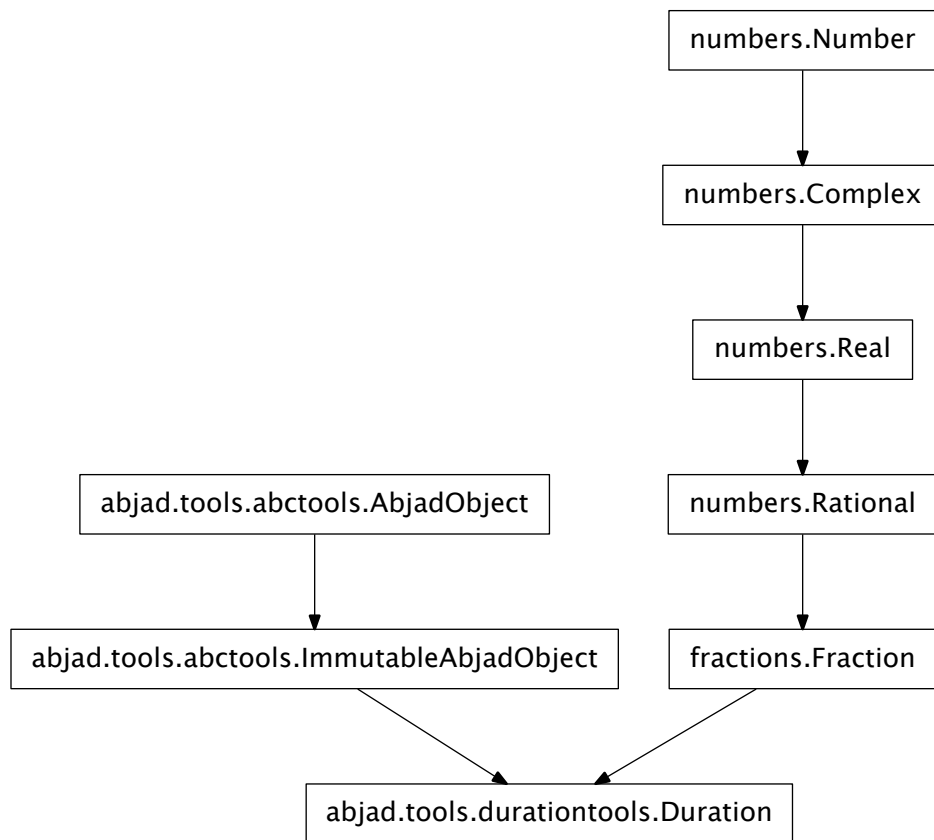
Skip nonsemantic contexts.

Function looks like a hack but isn't. LilyPond uses the dedicated command shown here to set accidental style. This means that it is not possible to set accidental style on a top-level context like score with a single override.

`durationtools`

concrete classes

`durationtools.Duration`



class `durationtools.Duration` (*args, **kwargs)
 New in version 2.0. Abjad model of musical duration.

Initialize from integer numerator:

```
>>> Duration(3)
Duration(3, 1)
```

Initialize from integer numerator and denominator:

```
>>> Duration(3, 16)
Duration(3, 16)
```

Initialize from integer-equivalent numeric numerator:

```
>>> Duration(3.0)
Duration(3, 1)
```

Initialize from integer-equivalent numeric numerator and denominator:

```
>>> Duration(3.0, 16)
Duration(3, 16)
```

Initialize from integer-equivalent singleton:

```
>>> Duration((3,))
Duration(3, 1)
```

Initialize from integer-equivalent pair:

```
>>> Duration((3, 16))
Duration(3, 16)
```

Initialize from other duration:

```
>>> Duration(Duration(3, 16))
Duration(3, 16)
```

Initialize from fraction:

```
>>> Duration(Fraction(3, 16))
Duration(3, 16)
```

Initialize from solidus string:

```
>>> Duration('3/16')
Duration(3, 16)
```

Changed in version 2.9: initialize from LilyPond duration string:

```
>>> Duration('8.')
Duration(3, 16)
```

Changed in version 2.9: initialize from nonreduced fraction:

```
>>> from abjad.tools import mathtools

>>> Duration(mathtools.NonreducedFraction(3, 16))
Duration(3, 16)
```

Durations inherit from built-in fraction:

```
>>> isinstance(Duration(3, 16), Fraction)
True
```

Durations are numeric:

```
>>> import numbers

>>> isinstance(Duration(3, 16), numbers.Number)
True
```

Durations are immutable.

Read-only Properties

`Duration.denominator`

Inherited from `fractions.Fraction`

`Duration.imag`

Real numbers have no imaginary component.

Inherited from `numbers.Real`

`Duration.numerator`

Inherited from `fractions.Fraction`

`Duration.pair`

New in version 2.9. Read-only pair of duration numerator and denominator:

```
>>> duration = Duration(3, 16)
```

```
>>> duration.pair
(3, 16)
```

Return integer pair.

`Duration.real`

Real numbers are their real component.

Inherited from `numbers.Real`

`Duration.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`Duration.conjugate()`

Conjugate is a no-op for Reals.

Inherited from `numbers.Real`

classmethod `Duration.from_decimal(dec)`

Converts a finite Decimal instance to a rational number, exactly.

Inherited from `fractions.Fraction`

classmethod `Duration.from_float(f)`

Converts a finite float to a rational number, exactly.

Beware that `Fraction.from_float(0.3) != Fraction(3, 10)`.

Inherited from `fractions.Fraction`

`Duration.limit_denominator(max_denominator=1000000)`

Closest Fraction to self with denominator at most `max_denominator`.

```
>>> Fraction('3.141592653589793').limit_denominator(10)
Fraction(22, 7)
>>> Fraction('3.141592653589793').limit_denominator(100)
Fraction(311, 99)
>>> Fraction(4321, 8765).limit_denominator(10000)
Fraction(4321, 8765)
```

Inherited from `fractions.Fraction`

Special Methods

`Duration.__abs__ (*args)`

`Duration.__add__ (*args)`

`Duration.__complex__ ()`
`complex(self) == complex(float(self), 0)`

Inherited from `numbers.Real`

`Duration.__div__ (*args)`

`Duration.__divmod__ (*args)`

`Duration.__eq__ (arg)`

`Duration.__float__ ()`
`float(self) = self.numerator / self.denominator`

It’s important that this conversion use the integer’s “true” division rather than casting one side to float before dividing so that ratios of huge integers convert without overflowing.

Inherited from `numbers.Rational`

`Duration.__floordiv__ (a, b)`
`a // b`

Inherited from `fractions.Fraction`

`Duration.__ge__ (arg)`

`Duration.__gt__ (arg)`

`Duration.__hash__ ()`
`hash(self)`

Tricky because values that are exactly representable as a float must have the same hash as that float.

Inherited from `fractions.Fraction`

`Duration.__le__ (arg)`

`Duration.__lt__ (arg)`

`Duration.__mod__ (*args)`

`Duration.__mul__ (*args)`

`Duration.__ne__ (arg)`

`Duration.__neg__ (*args)`

`Duration.__nonzero__ (a)`
`a != 0`

Inherited from `fractions.Fraction`

`Duration.__pos__ (*args)`

`Duration.__pow__ (*args)`

`Duration.__radd__ (*args)`

`Duration.__rdiv__ (*args)`

`Duration.__rdivmod__ (*args)`

`Duration.__repr__ ()`

Duration.__rfloordiv__(b, a)
a // b

Inherited from `fractions.Fraction`

Duration.__rmod__(*args)

Duration.__rmul__(*args)

Duration.__rpow__(*args)

Duration.__rsub__(*args)

Duration.__rtruediv__(*args)

Duration.__str__()
str(self)

Inherited from `fractions.Fraction`

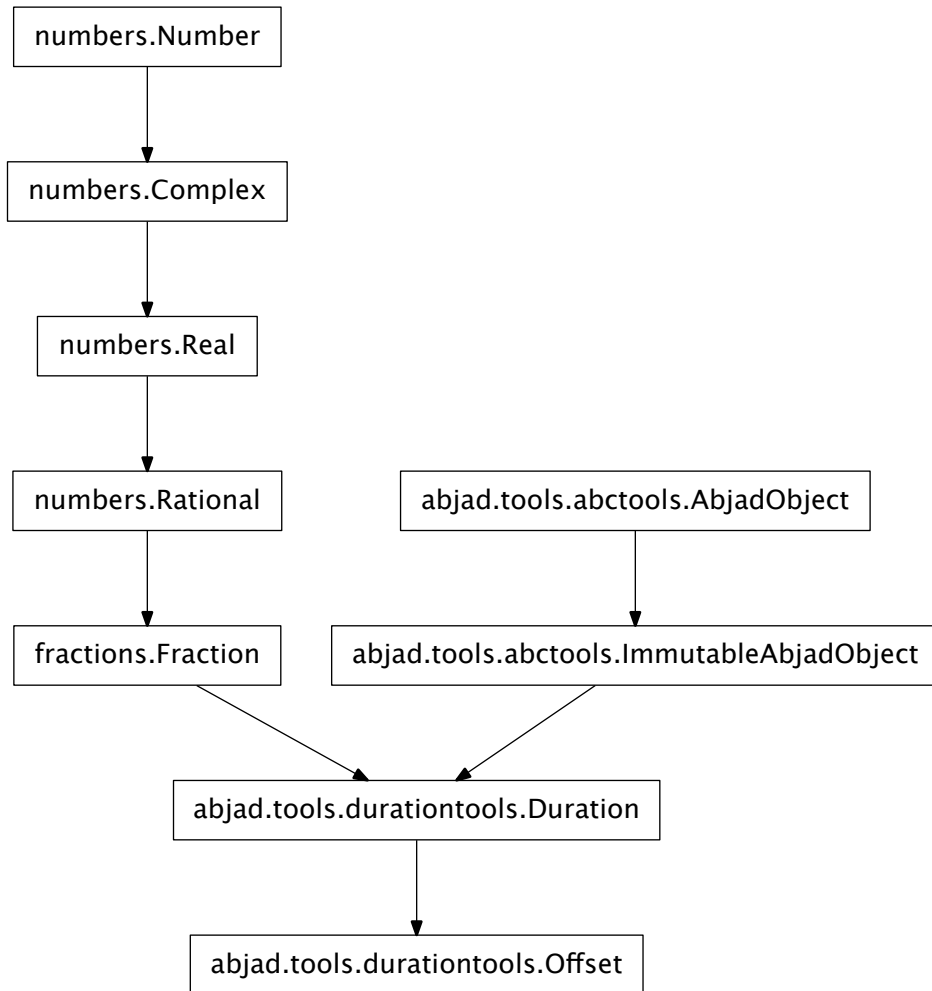
Duration.__sub__(*args)

Duration.__truediv__(*args)

Duration.__trunc__(a)
trunc(a)

Inherited from `fractions.Fraction`

durationtools.Offset



class `durationtools.Offset` (*args, **kwargs)
 New in version 2.0. Abjad model of offset value of musical time:

```
>>> from abjad.tools import durationtools
```

```
>>> durationtools.Offset(121, 16)
Offset(121, 16)
```

Offset inherits from duration (which inherits from built-in `Fraction`).

Read-only Properties

`Offset.denominator`

Inherited from `fractions.Fraction`

Offset.**imag**

Real numbers have no imaginary component.

Inherited from `numbers.Real`

Offset.**numerator**

Inherited from `fractions.Fraction`

Offset.**pair**

New in version 2.9. Read-only pair of duration numerator and denominator:

```
>>> duration = Duration(3, 16)
```

```
>>> duration.pair
(3, 16)
```

Return integer pair.

Inherited from `durationtools.Duration`

Offset.**real**

Real numbers are their real component.

Inherited from `numbers.Real`

Offset.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

Offset.**conjugate**()

Conjugate is a no-op for Reals.

Inherited from `numbers.Real`

classmethod Offset.**from_decimal**(*dec*)

Converts a finite Decimal instance to a rational number, exactly.

Inherited from `fractions.Fraction`

classmethod Offset.**from_float**(*f*)

Converts a finite float to a rational number, exactly.

Beware that `Fraction.from_float(0.3) != Fraction(3, 10)`.

Inherited from `fractions.Fraction`

Offset.**limit_denominator**(*max_denominator=1000000*)

Closest Fraction to self with denominator at most *max_denominator*.

```
>>> Fraction('3.141592653589793').limit_denominator(10)
Fraction(22, 7)
>>> Fraction('3.141592653589793').limit_denominator(100)
Fraction(311, 99)
>>> Fraction(4321, 8765).limit_denominator(10000)
Fraction(4321, 8765)
```

Inherited from `fractions.Fraction`

Special Methods

`Offset.__abs__(*args)`

Inherited from `durationontools.Duration`

`Offset.__add__(*args)`

Inherited from `durationontools.Duration`

`Offset.__complex__()`

`complex(self) == complex(float(self), 0)`

Inherited from `numbers.Real`

`Offset.__div__(*args)`

Inherited from `durationontools.Duration`

`Offset.__divmod__(*args)`

Inherited from `durationontools.Duration`

`Offset.__eq__(arg)`

Inherited from `durationontools.Duration`

`Offset.__float__()`

`float(self) = self.numerator / self.denominator`

It's important that this conversion use the integer's "true" division rather than casting one side to float before dividing so that ratios of huge integers convert without overflowing.

Inherited from `numbers.Rational`

`Offset.__floordiv__(a, b)`

`a // b`

Inherited from `fractions.Fraction`

`Offset.__ge__(arg)`

Inherited from `durationontools.Duration`

`Offset.__gt__(arg)`

Inherited from `durationontools.Duration`

`Offset.__hash__()`

`hash(self)`

Tricky because values that are exactly representable as a float must have the same hash as that float.

Inherited from `fractions.Fraction`

`Offset.__le__(arg)`

Inherited from `durationontools.Duration`

`Offset.__lt__(arg)`

Inherited from `durationontools.Duration`

`Offset.__mod__(*args)`

Inherited from `durationontools.Duration`

`Offset.__mul__(*args)`

Inherited from `durationontools.Duration`

`Offset.__ne__(arg)`

Inherited from `durationontools.Duration`

`Offset.__neg__(*args)`

Inherited from `durationontools.Duration`

`Offset.__nonzero__(a)`
`a != 0`

Inherited from `fractions.Fraction`

`Offset.__pos__(*args)`

Inherited from `durationtools.Duration`

`Offset.__pow__(*args)`

Inherited from `durationtools.Duration`

`Offset.__radd__(*args)`

Inherited from `durationtools.Duration`

`Offset.__rdiv__(*args)`

Inherited from `durationtools.Duration`

`Offset.__rdivmod__(*args)`

Inherited from `durationtools.Duration`

`Offset.__repr__()`

Inherited from `durationtools.Duration`

`Offset.__rfloordiv__(b, a)`

`a // b`

Inherited from `fractions.Fraction`

`Offset.__rmod__(*args)`

Inherited from `durationtools.Duration`

`Offset.__rmul__(*args)`

Inherited from `durationtools.Duration`

`Offset.__rpow__(*args)`

Inherited from `durationtools.Duration`

`Offset.__rsub__(*args)`

Inherited from `durationtools.Duration`

`Offset.__rtruediv__(*args)`

Inherited from `durationtools.Duration`

`Offset.__str__()`

`str(self)`

Inherited from `fractions.Fraction`

`Offset.__sub__(expr)`

New in version 2.10. Offset taken from offset returns duration:

```
>>> durationtools.Offset(2) - durationtools.Offset(1, 2)
Duration(3, 2)
```

Duration taken from offset returns another offset:

```
>>> durationtools.Offset(2) - durationtools.Duration(1, 2)
Offset(3, 2)
```

Coerce *expr* to offset when *expr* is neither offset nor duration:

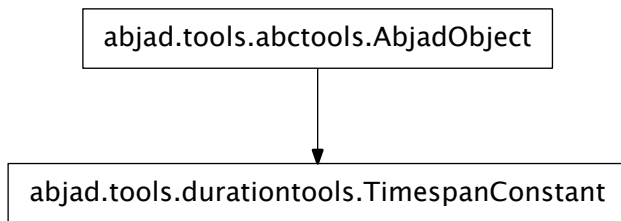
```
>>> durationtools.Offset(2) - Fraction(1, 2)
Duration(3, 2)
```

Return duration or offset.

`Offset.__truediv__(*args)`
 Inherited from `durationtools.Duration`

`Offset.__trunc__(a)`
`trunc(a)`
 Inherited from `fractions.Fraction`

`durationtools.TimespanConstant`



class `durationtools.TimespanConstant` (*start_offset=None, stop_offset=None*)
 New in version 1.0. Timespan constant.

Timespan constant `[1/2, 3/2]`:

```

>>> timespan_constant = durationtools.TimespanConstant((1, 2), (3, 2))

>>> timespan_constant
TimespanConstant(start_offset=Offset(1, 2), stop_offset=Offset(3, 2))

>>> z(timespan_constant)
durationtools.TimespanConstant(
    start_offset=durationtools.Offset(1, 2),
    stop_offset=durationtools.Offset(3, 2)
)
    
```

Timespan constants are object-modeled offset pairs.

Timespan constants are immutable.

Read-only Properties

`TimespanConstant.duration`
 Duration of timespan constant.

`TimespanConstant.start_offset`
 Start offset of timespan constant specified by user.
 Return offset.

`TimespanConstant.stop_offset`
 Stop offset of timespan constant specified by user.
 Return stop offset.

`TimespanConstant.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`TimespanConstant.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TimespanConstant.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimespanConstant.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TimespanConstant.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimespanConstant.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimespanConstant.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TimespanConstant.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

functions

`durationtools.all_are_duration_tokens`

`durationtools.all_are_duration_tokens(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad duration tokens:

```
>>> from abjad.tools import durationtools

>>> duration_tokens = ['8.', (3, 16), Fraction(3, 16), Duration(3, 16)]

>>> durationtools.all_are_duration_tokens(duration_tokens)
True
```

True when *expr* is an empty sequence:

```
>>> durationtools.all_are_duration_tokens([])
True
```

Otherwise false:

```
>>> durationtools.all_are_durations('foo')
False
```

Return boolean.

durationtools.all_are_durations

`durationtools.all_are_durations(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad durations:

```
>>> from abjad.tools import durationtools

>>> durations = [Duration((3, 16)), Duration((4, 16))]

>>> durationtools.all_are_durations(durations)
True
```

True when *expr* is an empty sequence:

```
>>> durationtools.all_are_durations([])
True
```

Otherwise false:

```
>>> durationtools.all_are_durations('foo')
False
```

Return boolean.

durationtools.assignable_rational_to_dot_count

`durationtools.assignable_rational_to_dot_count(rational)`

New in version 2.0. Change assignable *rational* to dot count:

```
>>> from abjad.tools import durationtools

>>> for n in range(1, 9):
...     try:
...         rational = Fraction(n, 16)
...         dot_count = durationtools.assignable_rational_to_dot_count(rational)
...         print '%s\t%s' % (rational, dot_count)
...     except AssignabilityError:
...         pass
```

```
...
1/16    0
1/8     0
3/16    1
1/4     0
3/8     1
7/16    2
1/2     0
```

Raise assignability error when *rational* not assignable.

Return nonnegative integer.

`durationtools.assignable_rational_to_lilypond_duration_string`

`durationtools.assignable_rational_to_lilypond_duration_string(rational)`

New in version 2.0. Change assignable *rational* to LilyPond duration string:

```
>>> from abjad.tools import durationtools

>>> durationtools.assignable_rational_to_lilypond_duration_string(Fraction(3, 16))
'8.'
```

Raise assignability error when *rational* not assignable.

Return string.

`durationtools.duration_pair_to_prolation_string`

`durationtools.duration_pair_to_prolation_string(pair)`

New in version 2.0. Change positive integer duration *pair* to colon-separated prolation string:

```
>>> from abjad.tools import durationtools

>>> durationtools.duration_pair_to_prolation_string((2, 3))
'3:2'
```

Return string.

`durationtools.duration_token_to_assignable_duration_pairs`

`durationtools.duration_token_to_assignable_duration_pairs(duration_token)`

New in version 1.1. Change *duration_token* to big-endian tuple of assignable duration pairs:

```
>>> from abjad.tools import durationtools

>>> duration_tokens = [(n, 16) for n in range(10, 20)]
>>> for duration_token in duration_tokens:
...     pairs = durationtools.duration_token_to_assignable_duration_pairs(
...         duration_token)
...     print duration_token, pairs
...
(10, 16) ((8, 16), (2, 16))
(11, 16) ((8, 16), (3, 16))
(12, 16) ((12, 16),)
(13, 16) ((12, 16), (1, 16))
```



```
(14, 16) ((14, 16),)
(15, 16) ((15, 16),)
(16, 16) ((16, 16),)
(17, 16) ((16, 16), (1, 16))
(18, 16) ((16, 16), (2, 16))
(19, 16) ((16, 16), (3, 16))
```

Return tuple of integer pairs. Changed in version 2.0: renamed `durationtools.token_decompose()` to `durationtools.duration_token_to_assignable_duration_pairs()`.

`durationtools.duration_token_to_duration_pair`

`durationtools.duration_token_to_duration_pair(duration_token)`

New in version 1.1. Change *duration_token* to duration pair:

```
>>> from abjad.tools import durationtools

>>> durationtools.duration_token_to_duration_pair(Fraction(2, 4))
(1, 2)
```

New in version 2.0: Change LilyPond duration string to duration pair:

```
>>> durationtools.duration_token_to_duration_pair('8.')
(3, 16)
```

Return pair. Changed in version 2.0: renamed `durationtools.token_unpack()` to `durationtools.duration_token_to_duration_pair()`.

`durationtools.duration_token_to_rational`

`durationtools.duration_token_to_rational(duration_token)`

New in version 2.0. Change *duration_token* to rational:

```
>>> from abjad.tools import durationtools

>>> durationtools.duration_token_to_rational((4, 16))
Fraction(1, 4)

>>> durationtools.duration_token_to_rational('4.')
Fraction(3, 8)
```

Return fraction.

`durationtools.duration_tokens_to_duration_pairs`

`durationtools.duration_tokens_to_duration_pairs(duration_tokens)`

New in version 2.0. Change *duration_tokens* to duration pairs:

```
>>> from abjad.tools import durationtools

>>> durationtools.duration_tokens_to_duration_pairs([Fraction(2, 4), 3, '8.', (5, 16)])
[(1, 2), (3, 1), (3, 16), (5, 16)]
```

Return new object of *duration_tokens* type.

durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator

`durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator` (*duration_tokens*)

New in version 2.0. Change *duration_tokens* to duration pairs with least common denominator:

```
>>> from abjad.tools import durationtools

>>> durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator(
...     [Fraction(2, 4), 3, '8.', (5, 16)])
[(8, 16), (48, 16), (3, 16), (5, 16)]
```

Return new object of *duration_tokens* type.

durationtools.duration_tokens_to_least_common_denominator

`durationtools.duration_tokens_to_least_common_denominator` (*duration_tokens*)

New in version 2.0. Change *duration_tokens* to least common denominator:

```
>>> from abjad.tools import durationtools

>>> durationtools.duration_tokens_to_least_common_denominator(
...     [Fraction(2, 4), 3, '8.', (5, 16)])
16
```

Return positive integer.

durationtools.duration_tokens_to_rationals

`durationtools.duration_tokens_to_rationals` (*duration_tokens*)

New in version 2.0. Change *duration_tokens* to rationals:

```
>>> from abjad.tools import durationtools

>>> durationtools.duration_tokens_to_rationals([Fraction(2, 4), 3, '8.', (5, 16)])
[Fraction(1, 2), Fraction(3, 1), Fraction(3, 16), Fraction(5, 16)]
```

Return new object of *duration_tokens* type.

durationtools.group_duration_tokens_by_implied_prolation

`durationtools.group_duration_tokens_by_implied_prolation` (*durations*)

New in version 1.1. Group *durations* by implied prolation:

```
>>> from abjad.tools import durationtools

>>> duration_tokens = [(1, 4), (1, 8), (1, 3), (1, 6), (1, 4)]

>>> durationtools.group_duration_tokens_by_implied_prolation(duration_tokens)
[[ (1, 4), (1, 8) ], [ (1, 3), (1, 6) ], [ (1, 4) ]]
```

Return list of integer pair lists. Changed in version 2.0: renamed `durationtools.agglomerate_by_prolation()` to `durationtools.group_duration_tokens_by_implied_prolation()`

durationtools.is_assignable_rational

`durationtools.is_assignable_rational(expr)`

New in version 1.1. True when *expr* is assignable rational. Otherwise false:

```
>>> from abjad.tools import durationtools

>>> for numerator in range(0, 16 + 1):
...     duration = Fraction(numerator, 16)
...     print '%s\t%s' % (duration, durationtools.is_assignable_rational(duration))
...
0      False
1/16   True
1/8    True
3/16   True
1/4    True
5/16   False
3/8    True
7/16   True
1/2    True
9/16   False
5/8    False
11/16  False
3/4    True
13/16  False
7/8    True
15/16  True
1      True
```

Return boolean. Changed in version 2.0: renamed `durationtools.is_assignable()` to `durationtools.is_assignable_rational()`.

durationtools.is_binary_rational

`durationtools.is_binary_rational(rational)`

New in version 1.1. True when *rational* is of the form $1/2**n$. Otherwise false:

```
>>> from abjad.tools import durationtools

>>> for n in range(1, 17):
...     rational = Fraction(1, n)
...     print '%s\t%s' % (rational, durationtools.is_binary_rational(rational))
...
1      True
1/2    True
1/3    False
1/4    True
1/5    False
1/6    False
1/7    False
1/8    True
1/9    False
1/10   False
1/11   False
1/12   False
1/13   False
1/14   False
```

```
1/15    False
1/16    True
```

Return boolean.

durationtools.is_duration_pair

`durationtools.is_duration_pair(arg)`

New in version 1.1. True when *arg* has the form of a pair of integers that initialize a positive rational:

```
>>> from abjad.tools import durationtools

>>> durationtools.is_duration_pair((5, 16))
True
```

Otherwise false:

```
>>> durationtools.is_duration_pair((-5, 16))
False
```

Return boolean. Changed in version 2.0: renamed `durationtools.is_pair()` to `durationtools.is_duration_pair()`.

durationtools.is_duration_token

`durationtools.is_duration_token(expr)`

New in version 2.0. True when *expr* has the form of an Abjad duration token:

```
>>> from abjad.tools import durationtools

>>> durationtools.is_duration_token('8.')
True
```

Otherwise false:

```
>>> durationtools.is_duration_token('foo')
False
```

Return boolean.

durationtools.is_lilypond_duration_name

`durationtools.is_lilypond_duration_name(expr)`

New in version 2.0. True when *expr* is a LilyPond duration name:

```
>>> from abjad.tools import durationtools

>>> durationtools.is_lilypond_duration_name('\\breve')
True
```

Otherwise false:

```
>>> durationtools.is_lilypond_duration_name('foo')
False
```

The regex `^(\\breve|\\longa|\\maxima)$` underlies this predicate.

Return boolean.

`durationtools.is_lilypond_duration_string`

`durationtools.is_lilypond_duration_string(expr)`

New in version 2.0. True when *expr* is a LilyPond duration string:

```
>>> from abjad.tools import durationtools

>>> durationtools.is_lilypond_duration_string('4.. * 1/2')
True
```

Otherwise false:

```
>>> durationtools.is_lilypond_duration_string('foo')
False
```

The regex `^(1|2|4|8|16|32|64|128|\\breve|\\longa|\\maxima)\\s*(\\.*)\\s*(*\\s*(\\d+(\\/\\d+)?)?)?$` underlies this predicate.

Return boolean.

`durationtools.is_proper_tuplet_multiplier`

`durationtools.is_proper_tuplet_multiplier(multiplier)`

True when $1/2 < multiplier < 2$.

```
>>> for n in range(17):
...     rational = Fraction(n, 8)
...     multiplier = durationtools.is_proper_tuplet_multiplier(rational)
...     print '%s %s' % (rational, multiplier)
...
0      False
1/8    False
1/4    False
3/8    False
1/2    False
5/8    True
3/4    True
7/8    True
1      True
9/8    True
5/4    True
11/8   True
3/2    True
13/8   True
7/4    True
15/8   True
2      False
```

This function models the idea that 4:3, 4:5, 4:6, 4:7 are valid tuplet multipliers while 4:2 and 4:8 aren't. Changed in version 2.0: renamed `durationtools.is_tuplet_multiplier()` to `durationtools.is_proper_tuplet_multiplier()`.

`durationtools.lilypond_duration_string_to_rational`

`durationtools.lilypond_duration_string_to_rational` (*duration_string*)

New in version 2.0. Change LilyPond *duration_string* to rational:

```
>>> from abjad.tools import durationtools

>>> durationtools.lilypond_duration_string_to_rational('8.')
Fraction(3, 16)
```

Return fraction.

`durationtools.lilypond_duration_string_to_rational_list`

`durationtools.lilypond_duration_string_to_rational_list` (*duration_string*)

New in version 2.0. Change LilyPond *duration_string* to rational list:

```
>>> from abjad.tools import durationtools

>>> durationtools.lilypond_duration_string_to_rational_list('8.. 32 8.. 32')
[Fraction(7, 32), Fraction(1, 32), Fraction(7, 32), Fraction(1, 32)]
```

Return list of fractions.

`durationtools.multiply_duration_pair`

`durationtools.multiply_duration_pair` (*pair*, *multiplier*)

New in version 1.1. Multiply duration *pair* by rational *multiplier*:

```
>>> from abjad.tools import durationtools

>>> durationtools.multiply_duration_pair((4, 8), Fraction(4, 5))
(16, 40)
```

Naive multiplication with no simplification of anything intended for certain types of meter multiplication.

Return integer pair. Changed in version 2.0: renamed `durationtools.pair_multiply_naive()` to `durationtools.multiply_duration_pair()`.

`durationtools.multiply_duration_pair_and_reduce_factors`

`durationtools.multiply_duration_pair_and_reduce_factors` (*pair*, *multiplier*)

New in version 1.1. Multiply *pair* by rational *multiplier* and reduce factors:

```
>>> from abjad.tools import durationtools

>>> durationtools.multiply_duration_pair_and_reduce_factors((4, 8), Fraction(2, 3))
(4, 12)
```

Intended for certain types of meter multiplication.

Return integer pair. Changed in version 2.0: renamed `durationtools.pair_multiply_reduce_factors()` to `durationtools.multiply_duration_pair_and_reduce_factors()`.

durationtools.multiply_duration_pair_and_try_to_preserve_numerator

`durationtools.multiply_duration_pair_and_try_to_preserve_numerator` (*pair*, *multiplier*)

New in version 1.1. Multiply duration *pair* by rational *multiplier* and try to preserve numerator:

```
>>> durationtools.multiply_duration_pair_and_try_to_preserve_numerator(
...     (9, 16), Fraction(2, 3))
(9, 24)
```

Intended for certain types of meter multiplication.

Return integer pair. Changed in version 2.0: renamed `durationtools.pair_multiply_constant_numerator()` to `durationtools.multiply_duration_pair_and_try_to_preserve_numerator()`.

durationtools.numeric_seconds_to_clock_string

`durationtools.numeric_seconds_to_clock_string` (*seconds*)

New in version 2.0. Change numeric *seconds* to clock string:

```
>>> from abjad.tools import durationtools

>>> durationtools.numeric_seconds_to_clock_string(117)
'1\57'
```

Return string.

durationtools.numeric_seconds_to_escaped_clock_string

`durationtools.numeric_seconds_to_escaped_clock_string` (*seconds*)

New in version 2.0. Change numeric *seconds* to escaped clock string:

```
>>> from abjad.tools import durationtools

>>> note = Note("c'4")
>>> clock_string = durationtools.numeric_seconds_to_escaped_clock_string(117)

>>> markuptools.Markup('"%s"' % clock_string, Up)(note)
Markup(('1\57\'',), direction=Up)(c'4)

>>> f(note)
c'4 ^ \markup { 1'57\' }
```

Escape seconds indicator for output as LilyPond markup.

Return string.

durationtools.positive_integer_to_implied_prolation_multiplier

`durationtools.positive_integer_to_implied_prolation_multiplier` (*n*)

New in version 1.1. Change positive integer *n* to implied porlation multiplier:

```
>>> for denominator in range(1, 17):
...     multiplier = durationtools.positive_integer_to_implied_prolation_multiplier(
...         denominator)
```

```
...     print '%s %4s' % (denominator, multiplier)
...
1         1
2         1
3         2/3
4         1
5         4/5
6         2/3
7         4/7
8         1
9         8/9
10        4/5
11        8/11
12        2/3
13        8/13
14        4/7
15        8/15
16        1
```

Return positive fraction less than or equal to 1. Changed in version 2.0: renamed `durationtools.denominator_to_multiplier()` to `durationtools.positive_integer_to IMPLIED_prolation_multiplier()`.

`durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator`

`durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator` (*durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator*)

Change *duration* to duration pair with multiple of specified *integer_denominator*:

```
>>> from abjad.tools import durationtools

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 2)
(1, 2)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 4)
(2, 4)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 8)
(4, 8)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 16)
(8, 16)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 3)
(3, 6)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 6)
(3, 6)
```



```
>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 12)
(6, 12)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 24)
(12, 24)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 5)
(5, 10)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 10)
(5, 10)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 20)
(10, 20)

>>> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(
...     Fraction(1, 2), 40)
(20, 40)
```

Return integer pair.

durationtools.rational_to_duration_pair_with_specified_integer_denominator

durationtools.rational_to_duration_pair_with_specified_integer_denominator (*duration*,
integer_denominator)

New in version 1.1. Change *duration* to duraiton pair with specified *integer_denominator*:

```
>>> from abjad.tools import durationtools

>>> for n in range(1, 17):
...     rational = Fraction(n, 16)
...     pair = durationtools.rational_to_duration_pair_with_specified_integer_denominator(
...         rational, 16)
...     print '%s\t%s' % (rational, pair)
...
1/16    (1, 16)
1/8     (2, 16)
3/16    (3, 16)
1/4     (4, 16)
5/16    (5, 16)
3/8     (6, 16)
7/16    (7, 16)
1/2     (8, 16)
9/16    (9, 16)
5/8     (10, 16)
11/16   (11, 16)
3/4     (12, 16)
13/16   (13, 16)
7/8     (14, 16)
```

```
15/16    (15, 16)
1         (16, 16)
```

Return integer pair. Changed in version 2.0: renamed `durationtools.in_terms_of()` to `durationtools.rational_to_duration_pair_with_specified_integer_denominator()`.

`durationtools.rational_to_equal_or_greater_assignable_rational`

`durationtools.rational_to_equal_or_greater_assignable_rational(rational)`

New in version 1.1. Change *rational* to equal or greater assignable rational:

```
>>> from abjad.tools import durationtools

>>> for n in range(1, 17):
...     prolated = Fraction(n, 16)
...     written = durationtools.rational_to_equal_or_greater_assignable_rational(prolated)
...     print '%s/16\t%s' % (n, written)
...
1/16    1/16
2/16    1/8
3/16    3/16
4/16    1/4
5/16    3/8
6/16    3/8
7/16    7/16
8/16    1/2
9/16    3/4
10/16   3/4
11/16   3/4
12/16   3/4
13/16   7/8
14/16   7/8
15/16   15/16
16/16   1
```

Return fraction.

Function returns dotted and double dotted durations where possible. Changed in version 2.0: Fixed to produce monotonically increasing output in response to monotonically increasing input. Changed in version 2.0: renamed `durationtools.prolated_to_written_not_less_than()` to `durationtools.rational_to_equal_or_greater_assignable_rational()`.

`durationtools.rational_to_equal_or_greater_binary_rational`

`durationtools.rational_to_equal_or_greater_binary_rational(rational)`

New in version 1.1. Change *rational* to equal to greater binary rational:

```
>>> for n in range(1, 17):
...     rational = Fraction(n, 16)
...     written_duration = durationtools.rational_to_equal_or_greater_binary_rational(
...         rational)
...     print '%s/16 %4s' % (n, written_duration)
...
1/16    1/16
2/16    1/8
3/16    1/4
```

```

4/16    1/4
5/16    1/2
6/16    1/2
7/16    1/2
8/16    1/2
9/16    1
10/16   1
11/16   1
12/16   1
13/16   1
14/16   1
15/16   1
16/16   1

```

```
>>> durationtools.rational_to_equal_or_greater_binary_rational(Fraction(1, 80))
Fraction(1, 64)
```

```
>>> durationtools.rational_to_equal_or_greater_binary_rational(Fraction(17, 16))
Fraction(2, 1)
```

Use to find written duration of tupletted leaves.

Return fraction. Changed in version 2.0: renamed `durationtools.naive_prolated_to_written_not_less_than` to `durationtools.rational_to_equal_or_greater_binary_rational()`.

`durationtools.rational_to_equal_or_lesser_assignable_rational`

`durationtools.rational_to_equal_or_lesser_assignable_rational(rational)`

New in version 1.1. Change *rational* to equal or lesser assignable rational:

```
>>> from abjad.tools import durationtools

>>> for n in range(1, 17):
...     rational = Fraction(n, 16)
...     written = durationtools.rational_to_equal_or_lesser_assignable_rational(rational)
...     print '%s/16\t%s' % (n, written)
...
1/16    1/16
2/16    1/8
3/16    3/16
4/16    1/4
5/16    1/4
6/16    3/8
7/16    7/16
8/16    1/2
9/16    1/2
10/16   1/2
11/16   1/2
12/16   3/4
13/16   3/4
14/16   7/8
15/16   15/16
16/16   1

```

Return fraction.

Function returns dotted and double dotted durations where possible. Changed in version 2.0: Fixed to produce monotonically increasing output in response to monotonically increasing input.Changed

in version 2.0: renamed `durationtools.prolated_to_written_not_greater_than()` to `durationtools.rational_to_equal_or_lesser_assignable_rational()`.

`durationtools.rational_to_equal_or_lesser_binary_rational`

`durationtools.rational_to_equal_or_lesser_binary_rational(rational)`

New in version 1.1. Change *rational* to equal or lesser binary rational:

```
>>> for n in range(1, 17):
...     rational = Fraction(n, 16)
...     written_duration = durationtools.rational_to_equal_or_lesser_binary_rational(
...         rational)
...     print '%s/16 %4s' % (n, written_duration)
...
1/16    1/16
2/16    1/8
3/16    1/8
4/16    1/4
5/16    1/4
6/16    1/4
7/16    1/4
8/16    1/2
9/16    1/2
10/16   1/2
11/16   1/2
12/16   1/2
13/16   1/2
14/16   1/2
15/16   1/2
16/16   1

>>> durationtools.rational_to_equal_or_lesser_binary_rational(Fraction(1, 80))
Fraction(1, 128)
```

Return fraction.

Function intended to find written duration of notes inside tuplet. Changed in version 2.0: re-named `durationtools.naive_prolated_to_written_not_greater_than()` to `durationtools.rational_to_equal_or_lesser_binary_rational()`.

`durationtools.rational_to_flag_count`

`durationtools.rational_to_flag_count(rational)`

New in version 2.0. Change *rational* to number of flags required to notate:

```
>>> from abjad.tools import durationtools

>>> durationtools.rational_to_flag_count(Fraction(1, 32))
3
```

Return nonnegative integer.

`durationtools.rational_to_fraction_string`

`durationtools.rational_to_fraction_string(rational)`

New in version 1.1. Change *rational* to fraction string:

```
>>> from abjad.tools import durationtools

>>> durationtools.rational_to_fraction_string(Fraction(2, 4))
'1/2'
```

Return string.

`durationtools.rational_to_prolation_string`

`durationtools.rational_to_prolation_string(rational)`

New in version 2.0. Change *rational* to prolation string:

```
>>> from abjad.tools import durationtools

>>> generator = durationtools.yield_all_positive_rationals_uniquely()
>>> for n in range(16):
...     rational = generator.next()
...     prolation_string = durationtools.rational_to_prolation_string(rational)
...     print '%s\\t%s' % (rational, prolation_string)
...
1          1:1
2          1:2
1/2        2:1
1/3        3:1
3          1:3
4          1:4
3/2        2:3
2/3        3:2
1/4        4:1
1/5        5:1
5          1:5
6          1:6
5/2        2:5
4/3        3:4
3/4        4:3
2/5        5:2
```

Return string.

`durationtools.rational_to_proper_fraction`

`durationtools.rational_to_proper_fraction(rational)`

New in version 2.0. Change *rational* to proper fraction:

```
>>> from abjad.tools import durationtools

>>> durationtools.rational_to_proper_fraction(Fraction(116, 8))
(14, Fraction(1, 2))
```

Return pair.

durationtools.rewrite_rational_under_new_tempo

`durationtools.rewrite_rational_under_new_tempo` (*prolated_duration_1*, *tempo_mark_1*,
tempo_mark_2)

New in version 2.0. Given *prolated_duration_1* governed by *tempo_mark_1*, return *prolated_duration_2* governed by *tempo_mark_2* such that *prolated_duration_1* and *prolated_duration_2* consume exactly the same amount of time in seconds.

Consider the two tempo indications below.

```
>>> from abjad.tools import durationtools

>>> tempo_mark_1 = contexttools.TempoMark(Duration(1, 4), 60)
>>> tempo_mark_2 = contexttools.TempoMark(Duration(1, 4), 90)
```

The first tempo indication specifies quarter = 60 MM. The second tempo indication specifies quarter = 90 MM.

The second tempo is 1 1/2 times as fast as the first:

```
>>> tempo_mark_2 / tempo_mark_1
Duration(3, 2)
```

An triplet eighth note at tempo 1 equals a regular eighth note at tempo 2.

```
>>> durationtools.rewrite_rational_under_new_tempo(
...     Duration(1, 12), tempo_mark_1, tempo_mark_2)
Duration(1, 8)
```

Conversely, a regular eighth note at tempo 1 equals a dotted sixteenth at tempo 2.

```
>>> durationtools.rewrite_rational_under_new_tempo(
...     Duration(1, 8), tempo_mark_1, tempo_mark_2)
Duration(3, 16)
```

Return fraction.

durationtools.yield_all_assignable_rationals

`durationtools.yield_all_assignable_rationals` ()

New in version 2.0. Yield all assignable rationals in Cantor diagonalized order:

```
>>> from abjad.tools import durationtools

>>> generator = durationtools.yield_all_assignable_rationals()
>>> for n in range(16):
...     generator.next()
...
Fraction(1, 1)
Fraction(2, 1)
Fraction(1, 2)
Fraction(3, 1)
Fraction(4, 1)
Fraction(3, 2)
Fraction(1, 4)
Fraction(6, 1)
Fraction(3, 4)
Fraction(7, 1)
Fraction(8, 1)
Fraction(7, 2)
```

```
Fraction(1, 8)
Fraction(7, 4)
Fraction(3, 8)
Fraction(12, 1)
```

Return fraction generator.

`durationtools.yield_all_positive_integer_pairs`

`durationtools.yield_all_positive_integer_pairs()`

New in version 2.0. Yield all positive integer pairs in Cantor diagonalized order:

```
>>> from abjad.tools import durationtools

>>> generator = durationtools.yield_all_positive_integer_pairs()
>>> for n in range(16):
...     generator.next()
...
(1, 1)
(2, 1)
(1, 2)
(1, 3)
(2, 2)
(3, 1)
(4, 1)
(3, 2)
(2, 3)
(1, 4)
(1, 5)
(2, 4)
(3, 3)
(4, 2)
(5, 1)
(6, 1)
```

Return pair generator.

`durationtools.yield_all_positive_rationals`

`durationtools.yield_all_positive_rationals()`

New in version 2.0. Yield all positive rationals in Cantor diagonalized order:

```
>>> from abjad.tools import durationtools

>>> generator = durationtools.yield_all_positive_rationals()
>>> for n in range(16):
...     generator.next()
...
Fraction(1, 1)
Fraction(2, 1)
Fraction(1, 2)
Fraction(1, 3)
Fraction(1, 1)
Fraction(3, 1)
Fraction(4, 1)
Fraction(3, 2)
```

```
Fraction(2, 3)
Fraction(1, 4)
Fraction(1, 5)
Fraction(1, 2)
Fraction(1, 1)
Fraction(2, 1)
Fraction(5, 1)
Fraction(6, 1)
```

Return fraction generator.

durationtools.yield_all_positive_rationals_uniquely

`durationtools.yield_all_positive_rationals_uniquely()`

New in version 2.0. Yield all positive rationals in Cantor diagonalized order uniquely:

```
>>> generator = durationtools.yield_all_positive_rationals_uniquely()
>>> for n in range(16):
...     generator.next()
...
Fraction(1, 1)
Fraction(2, 1)
Fraction(1, 2)
Fraction(1, 3)
Fraction(3, 1)
Fraction(4, 1)
Fraction(3, 2)
Fraction(2, 3)
Fraction(1, 4)
Fraction(1, 5)
Fraction(5, 1)
Fraction(6, 1)
Fraction(5, 2)
Fraction(4, 3)
Fraction(3, 4)
Fraction(2, 5)
```

Return fraction generator.

durationtools.yield_prolation_rewrite_pairs

`durationtools.yield_prolation_rewrite_pairs(prolated_duration, mini-
mum_written_duration=None)`

New in version 2.0. Yield all prolation rewrite pairs of *prolated_duration* in Cantor diagonalized order.

Ensure written duration never less than *minimum_written_duration*.

The different ways to notate a prolated duration of 1/8:

```
>>> from abjad.tools import durationtools

>>> pairs = durationtools.yield_prolation_rewrite_pairs(
...     Fraction(1, 8))

>>> for pair in pairs: pair
...
(Fraction(1, 1), Fraction(1, 8))
```



```
(Fraction(2, 3), Fraction(3, 16))
(Fraction(4, 3), Fraction(3, 32))
(Fraction(4, 7), Fraction(7, 32))
(Fraction(8, 7), Fraction(7, 64))
(Fraction(8, 15), Fraction(15, 64))
(Fraction(16, 15), Fraction(15, 128))
(Fraction(16, 31), Fraction(31, 128))
```

The different ways to notate a prolated duration of 1/12:

```
>>> pairs = durationtools.yield_prolation_rewrite_pairs(
... Fraction(1, 12))

>>> for pair in pairs: pair
...
(Fraction(2, 3), Fraction(1, 8))
(Fraction(4, 3), Fraction(1, 16))
(Fraction(8, 9), Fraction(3, 32))
(Fraction(16, 9), Fraction(3, 64))
(Fraction(16, 21), Fraction(7, 64))
(Fraction(32, 21), Fraction(7, 128))
(Fraction(32, 45), Fraction(15, 128))
```

The different ways to notate a prolated duration of 5/48:

```
>>> pairs = durationtools.yield_prolation_rewrite_pairs(
... Fraction(5, 48))

>>> for pair in pairs: pair
...
(Fraction(5, 6), Fraction(1, 8))
(Fraction(5, 3), Fraction(1, 16))
(Fraction(5, 9), Fraction(3, 16))
(Fraction(10, 9), Fraction(3, 32))
(Fraction(20, 21), Fraction(7, 64))
(Fraction(40, 21), Fraction(7, 128))
(Fraction(8, 9), Fraction(15, 128))
```

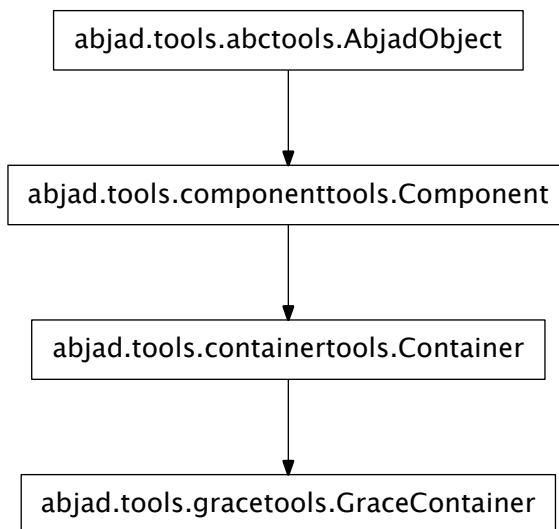
When *minimum_written_duration* is none set to 1/128.

Return generator of paired fractions.

gracetools

concrete classes

gracetools.GraceContainer



class gracetools.**GraceContainer** (*music=None, kind='grace', **kwargs*)

Abjad model of grace music:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(voice[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(voice)
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}

>>> grace_notes = [Note("c'16"), Note("d'16")]
>>> gracetools.GraceContainer(grace_notes, kind='grace')(voice[1])
Note("d'8")

>>> f(voice)
\new Voice {
  c'8 [
    \grace {
      c'16
      d'16
    }
  ]
}
  
```

```

    }
    d'8
    e'8
    f'8 ]
}

>>> after_grace_notes = [Note("e'16"), Note("f'16")]
>>> gracetools.GraceContainer(after_grace_notes, kind='after')(voice[1])
Note("d'8")

>>> f(voice)
\new Voice {
  c'8 [
    \grace {
      c'16
      d'16
    }
    \afterGrace
    d'8
    {
      e'16
      f'16
    }
  ]
  e'8
  f'8 ]
}

```

Grace objects are containers you can fill with notes, rests and chords.

Grace containers override the special `__call__` method.

Use `GraceContainer()` to attach grace containers to nongrace notes, rests and chords.

Read-only Properties

`GraceContainer.contents_duration`

Inherited from `containertools.Container`

`GraceContainer.duration_in_seconds`

Inherited from `containertools.Container`

`GraceContainer.leaves`

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

`GraceContainer.lilypond_format`

`GraceContainer.music`

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

`GraceContainer.override`

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

`GraceContainer.parent`

Inherited from `componenttools.Component`

`GraceContainer.preprolated_duration`

Inherited from `containertools.Container`

`GraceContainer.prolated_duration`

Inherited from `componenttools.Component`

`GraceContainer.prolation`

Inherited from `componenttools.Component`

`GraceContainer.set`

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

`GraceContainer.spanners`

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`GraceContainer.start_offset`

Read-only start offset of component.

Inherited from `componenttools.Component`

`GraceContainer.start_offset_in_seconds`

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

`GraceContainer.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`GraceContainer.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`GraceContainer.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties**GraceContainer.is_parallel**

Get parallel container:

```

>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}

>>> container.is_parallel
False

```

Return boolean.

Set parallel container:

```

>>> container.is_parallel = True

>>> f(container)
<<
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
>>

```

Return none.

Inherited from `containertools.Container`**GraceContainer.kind**Get *kind* of grace container:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> gracetools.GraceContainer([Note("cs'16")], kind = 'grace')(staff[1])
Note("d'8")
>>> grace_container = staff[1].grace
>>> grace_container.kind
'grace'

```

Return string.

Set *kind* of grace container:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> gracetools.GraceContainer([Note("cs'16")], kind = 'grace')(staff[1])
Note("d'8")
>>> grace_container = staff[1].grace
>>> grace_container.kind = 'acciaccatura'

```

```
>>> grace_container.kind
'acciaccatura'
```

Set string.

Valid options include 'after', 'grace', 'acciaccatura', 'appoggiatura'.

Methods

GraceContainer.**append**(*component*)

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

GraceContainer.**detach**()

Detach grace container from leaf:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> grace_container = gracetools.GraceContainer([Note("cs'16")], kind = 'grace')
>>> grace_container(staff[1])
Note("d'8")
>>> f(staff)
\new Staff {
    c'8
    \grace {
        cs'16
    }
    d'8
    e'8
    f'8
}

>>> grace_container.detach()
GraceContainer()
>>> f(staff)
\new Staff {
    c'8
    d'8
```

```

    e'8
    f'8
}

```

Return grace container.

`GraceContainer.extend(expr)`

Extend *expr* against container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```

Return none. New in version 2.3: *expr* may now be a LilyPond input string. Inherited from `containertools.Container`

`GraceContainer.index(component)`

Index *component* in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2

```

Return nonnegative integer.

Inherited from `containertools.Container`

`GraceContainer.insert(i, component)`

Insert *component* in container at index *i*:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8

```

```

        e'8 ]
    }

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}

```

Return none.

Inherited from `containertools.Container`

`GraceContainer.pop(i=-1)`

Pop component at index *i* from container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}

```

Return component.

Inherited from `containertools.Container`

`GraceContainer.remove(component)`

Remove *component* from container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

```



```
>>> container.remove(note)

>>> f(container)
{
  c'8 [
  d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

`GraceContainer.__add__(expr)`

Concatenate containers self and expr. The operation $c = a + b$ returns a new Container c with the content of both a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

`GraceContainer.__call__(arg)`

`GraceContainer.__contains__(expr)`

True if expr is in container, otherwise False.

Inherited from `containertools.Container`

`GraceContainer.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

`GraceContainer.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`GraceContainer.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GraceContainer.__getitem__(i)`

Return component at index i in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`GraceContainer.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`GraceContainer.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`GraceContainer.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

`GraceContainer.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GraceContainer.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`GraceContainer.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GraceContainer.__mul__(n)`

Inherited from `componenttools.Component`

`GraceContainer.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`GraceContainer.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`GraceContainer.__repr__()`

`GraceContainer.__rmul__(n)`

Inherited from `componenttools.Component`

`GraceContainer.__setitem__(i, expr)`

Set ‘`expr`’ in self at nonnegative integer index `i`. Or, set ‘`expr`’ in self at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with ‘`expr`’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

functions

`gracetools.all_are_grace_containers`

`gracetools.all_are_grace_containers(expr, kind=None)`

New in version 2.6. True when `expr` is a sequence of Abjad grace containers:

```
>>> graces = [
...     gracetools.GraceContainer("<c' e' g'>4"),
...     gracetools.GraceContainer("<c' f' a'>4")]
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> grace_notes = [Note("c'16"), Note("d'16")]
```

```

>>> grace_container = gracetools.GraceContainer(grace_notes, kind='grace')
>>> grace_container(voice[1])
Note("d'8")

>>> f(voice)
\new Voice {
  c'8
  \grace {
    c'16
    d'16
  }
  d'8
  e'8
  f'8
}

>>> gracetools.all_are_grace_containers([grace_container])
True

```

True when *expr* is an empty sequence:

```

>>> gracetools.all_are_grace_containers([])
True

```

Otherwise false:

```

>>> gracetools.all_are_grace_containers('foo')
False

```

Return boolean.

gracetools.detach_grace_containers_attached_to_leaf

gracetools.detach_grace_containers_attached_to_leaf (*leaf*, *kind=None*)

New in version 2.0. Detach grace containers attached to *leaf*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> grace_container = gracetools.GraceContainer([Note("cs'16")], kind='grace')
>>> grace_container(staff[1])
Note("d'8")

>>> f(staff)
\new Staff {
  c'8
  \grace {
    cs'16
  }
  d'8
  e'8
  f'8
}

>>> gracetools.get_grace_containers_attached_to_leaf(staff[1])
(GraceContainer(cs'16),)

>>> gracetools.detach_grace_containers_attached_to_leaf(staff[1])
(GraceContainer(),)

```

```
>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

>>> gracetools.get_grace_containers_attached_to_leaf(staff[1])
()
```

Return tuple.

gracetools.detach_grace_containers_attached_to_leaves_in_expr

gracetools.detach_grace_containers_attached_to_leaves_in_expr(*expr*,
kind=None)

New in version 2.9. Detach grace containers attached to leaves in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> grace_container = gracetools.GraceContainer([Note("cs'16")], kind='grace')
>>> grace_container(staff[1])
Note("d'8")

>>> f(staff)
\new Staff {
    c'8
    \grace {
        cs'16
    }
    d'8
    e'8
    f'8
}

>>> gracetools.detach_grace_containers_attached_to_leaves_in_expr(staff)
(GraceContainer(),)

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
```

Return tuple of zero or more grace containers.

gracetools.get_grace_containers_attached_to_leaf

gracetools.get_grace_containers_attached_to_leaf(*leaf*, kind=None)

New in version 2.0. Example 1. Get all grace containers attached to *leaf*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> gracetools.GraceContainer([Note("cs'16")], kind='grace')(staff[1])
Note("d'8")
```

```
>>> gracetools.GraceContainer([Note("ds'16")], kind='after')(staff[1])
Note("d'8")
```

```
>>> f(staff)
\new Staff {
  c'8
  \grace {
    cs'16
  }
  \afterGrace
  d'8
  {
    ds'16
  }
  e'8
  f'8
}
```

```
>>> gracetools.get_grace_containers_attached_to_leaf(staff[1])
(GraceContainer(cs'16), GraceContainer(ds'16))
```

Example 2. Get only (proper) grace containers attached to *leaf*:

```
>>> gracetools.get_grace_containers_attached_to_leaf(staff[1], kind='grace')
(GraceContainer(cs'16),)
```

Example 3. Get only after grace containers attached to *leaf*:

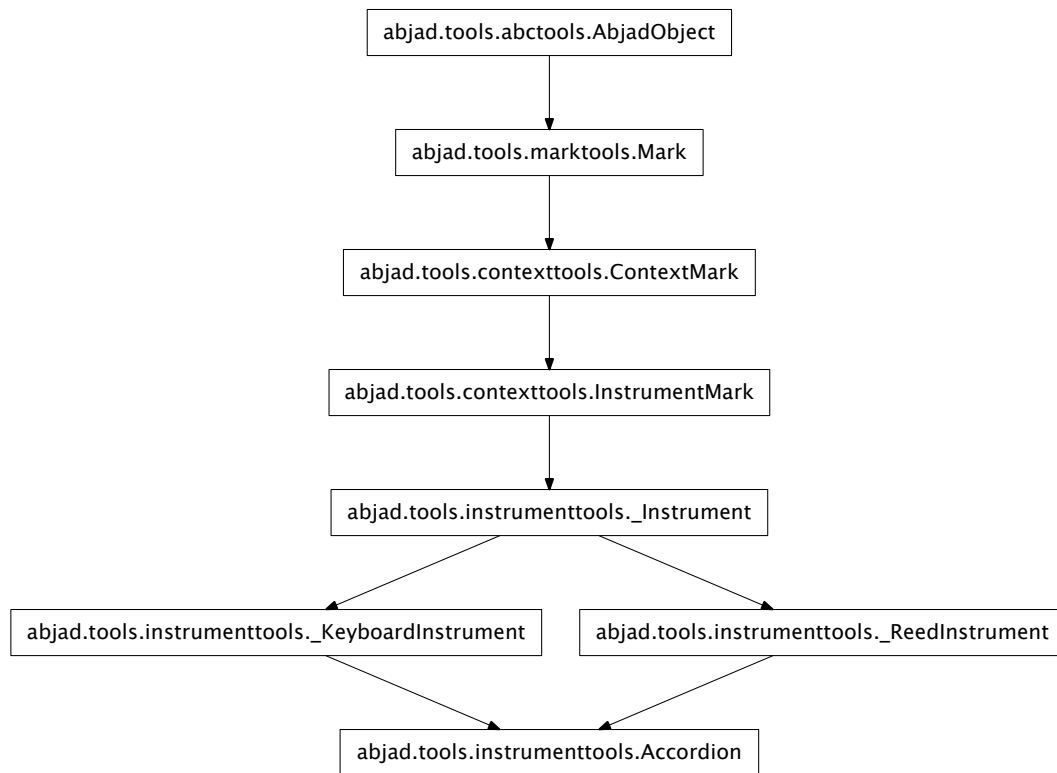
```
>>> gracetools.get_grace_containers_attached_to_leaf(staff[1], kind='after')
(GraceContainer(ds'16),)
```

Return tuple.

instrumenttools

concrete classes

instrumenttools.Accordion



class instrumenttools.**Accordion**(*target_context=None*, ***kwargs*)

Abjad model of the accordion:

```
>>> piano_staff = scoretools.PianoStaff([Staff("c'8 d'8 e'8 f'8"), Staff("c'4 b4")])
```

```
>>> instrumenttools.Accordion()(piano_staff)
```

```
Accordion() (PianoStaff<<2>>)
```

```
>>> f(piano_staff)
```

```
\new PianoStaff <<
  \set PianoStaff.instrumentName = \markup { Accordion }
  \set PianoStaff.shortInstrumentName = \markup { Acc. }
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
```

```

    \new Staff {
      c'4
      b4
    }
  >>

```

The accordion targets piano staff context by default.

Read-only Properties

Accordion.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Accordion.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Accordion.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Accordion.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Accordion.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Accordion.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Accordion.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Accordion.lilypond_format

Read-only LilyPond input format of instrument mark:

```

>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Inherited from `contexttools.InstrumentMark`

Accordion.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Accordion.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Accordion.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Accordion.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Accordion.all_clefs

Inherited from `instrumenttools._Instrument`

Accordion.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Accordion.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Accordion.pitch_range

Inherited from instrumenttools._Instrument

Accordion.primary_clefs

Inherited from instrumenttools._Instrument

Accordion.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Accordion.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Accordion.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

`Accordion.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`Accordion.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`Accordion.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`Accordion.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`Accordion.__call__(*args)`

Inherited from `marktools.Mark`

`Accordion.__delattr__(*args)`

Inherited from `marktools.Mark`

`Accordion.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`Accordion.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Accordion.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Accordion.__hash__()`

Inherited from `contexttools.InstrumentMark`

`Accordion.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Accordion.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

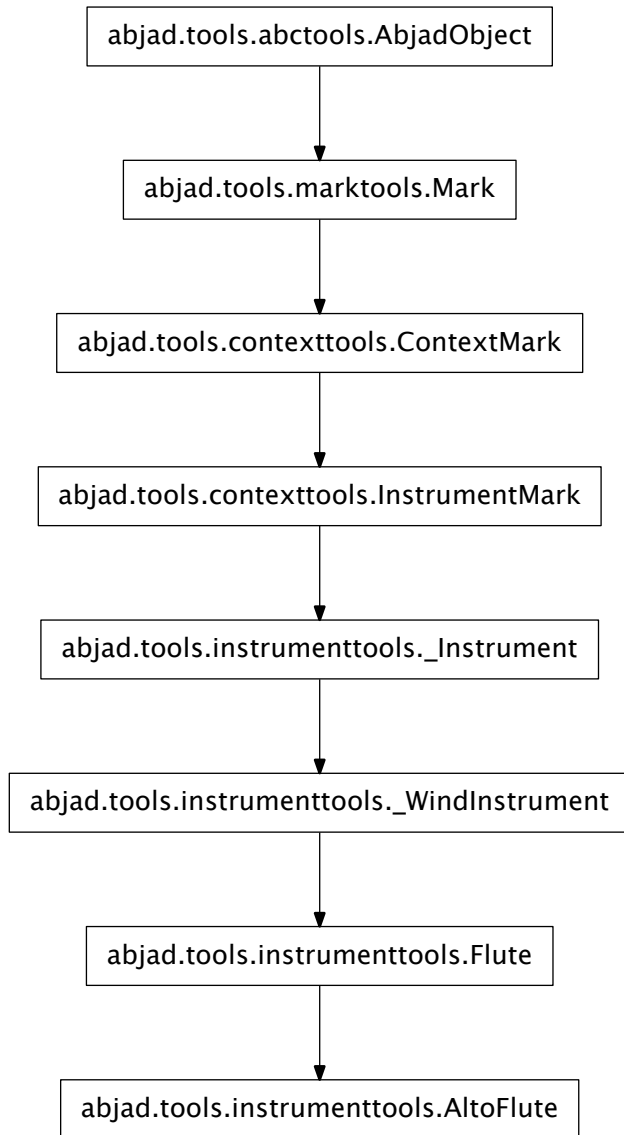
`Accordion.__ne__(arg)`

Inherited from `marktools.Mark`

`Accordion.__repr__()`

Inherited from `marktools.Mark`

instrumenttools.Alt flute



```

class instrumenttools.Alt flute(**kwargs)
    Abjad model of the alto flute:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.Alt flute()(staff)
    Alt flute()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Alto flute }
  \set Staff.shortInstrumentName = \markup { Alt. fl. }
  c'8
  d'8
  e'8
  f'8
}
```

The alto flute targets staff context by default.

Read-only Properties

AltoFlute.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

AltoFlute.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

AltoFlute.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

AltoFlute.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

AltoFlute.is_primary_instrument

Inherited from `instrumenttools._Instrument`

AltoFlute.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

AltoFlute.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

AltoFlute.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

AltoFlute.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

AltoFlute.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

AltoFlute.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

AltoFlute.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

AltoFlute.all_clefs

Inherited from `instrumenttools._Instrument`

AltoFlute.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`AltoFlute.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`AltoFlute.pitch_range`

Inherited from `instrumenttools._Instrument`

`AltoFlute.primary_clefs`

Inherited from `instrumenttools._Instrument`

`AltoFlute.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`AltoFlute.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`AltoFlute.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`AltoFlute.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`AltoFlute.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`AltoFlute.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`AltoFlute.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`AltoFlute.__call__(*args)`

Inherited from `marktools.Mark`

`AltoFlute.__delattr__(*args)`

Inherited from `marktools.Mark`

`AltoFlute.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`AltoFlute.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AltoFlute.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`


```

AltoFlute.__hash__()
    Inherited from contexttools.InstrumentMark

AltoFlute.__le__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject

AltoFlute.__lt__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

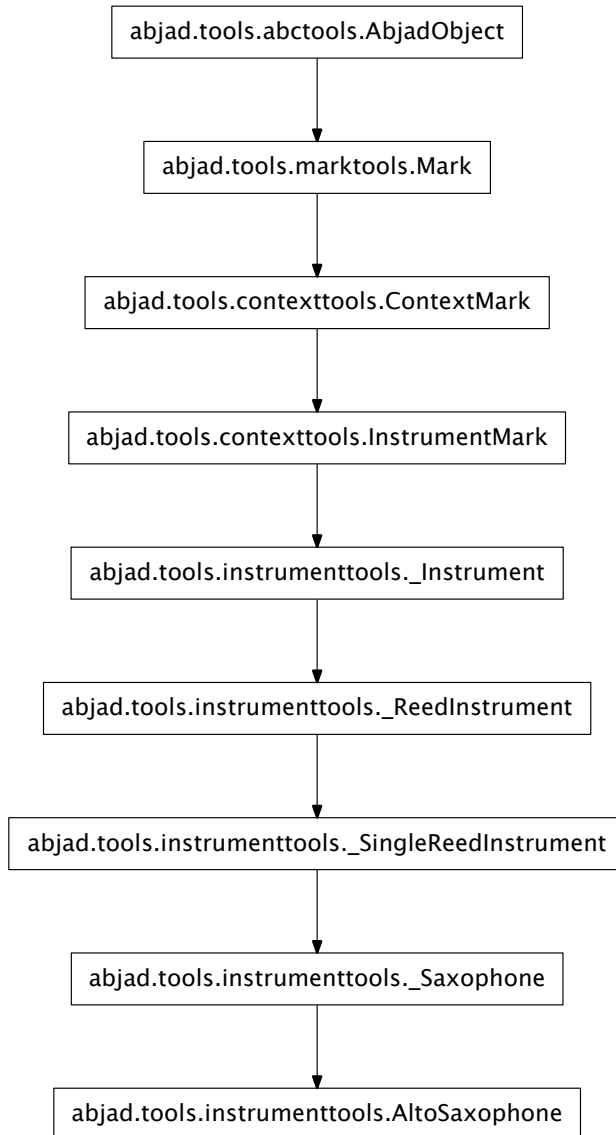
    Inherited from abctools.AbjadObject

AltoFlute.__ne__(arg)
    Inherited from marktools.Mark

AltoFlute.__repr__()
    Inherited from marktools.Mark

```

instrumenttools.AltoSaxophone



```

class instrumenttools.AltoSaxophone(**kwargs)
    New in version 2.6. Abjad model of the alto saxophone:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.AltoSaxophone()(staff)
    AltoSaxophone()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Alto saxophone }
  \set Staff.shortInstrumentName = \markup { Alto sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The alto saxophone is pitched in E-flat.

The alto saxophone targets staff context by default.

Read-only Properties

`AltoSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`AltoSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

`AltoSaxophone.is_primary_instrument`

Inherited from `instrumenttools._Instrument`

`AltoSaxophone.is_secondary_instrument`

Inherited from `instrumenttools._Instrument`

`AltoSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

`AltoSaxophone.lilypond_format`

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`AltoSaxophone.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`AltoSaxophone.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`AltoSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

`AltoSaxophone.all_clefs`

Inherited from `instrumenttools._Instrument`

`AltoSaxophone.instrument_name`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

AltoSaxophone.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

AltoSaxophone.pitch_range

Inherited from `instrumenttools._Instrument`

AltoSaxophone.primary_clefs

Inherited from `instrumenttools._Instrument`

AltoSaxophone.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

AltoSaxophone.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`AltoSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`AltoSaxophone.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`AltoSaxophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`AltoSaxophone.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`AltoSaxophone.__call__(*args)`

Inherited from `marktools.Mark`

`AltoSaxophone.__delattr__(*args)`

Inherited from `marktools.Mark`

`AltoSaxophone.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AltoSaxophone.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`AltoSaxophone.__hash__()`
Inherited from `contexttools.InstrumentMark`

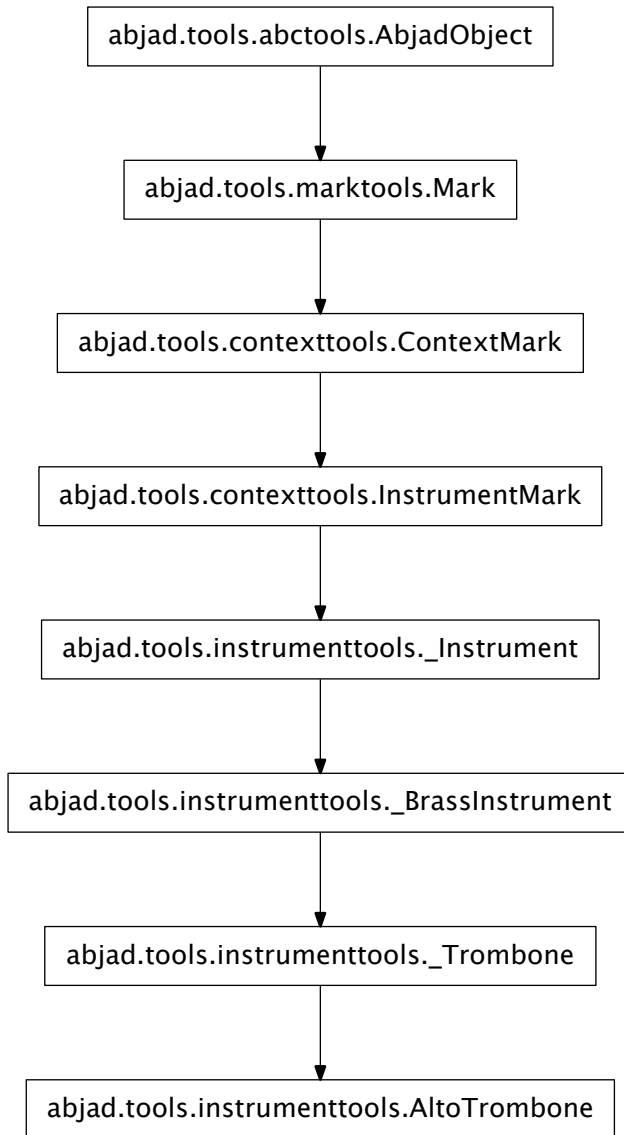
`AltoSaxophone.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`AltoSaxophone.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`AltoSaxophone.__ne__(arg)`
Inherited from `marktools.Mark`

`AltoSaxophone.__repr__()`
Inherited from `marktools.Mark`

instrumenttools.AltiTrombone



class instrumenttools.**AltoTrombone** (**kwargs)

New in version 2.0. Abjad model of the alto trombone:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

>>> instrumenttools.AltiTrombone()(staff)
AltoTrombone()(Staff{4})
  
```



```
>>> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Alto trombone }
  \set Staff.shortInstrumentName = \markup { Alt. trb. }
  c'8
  d'8
  e'8
  f'8
}
```

The tenor trombone targets staff context by default.

Read-only Properties

AltoTrombone.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

AltoTrombone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

AltoTrombone.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

AltoTrombone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

AltoTrombone.is_primary_instrument

Inherited from `instrumenttools._Instrument`

AltoTrombone.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

AltoTrombone.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

AltoTrombone.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

AltoTrombone.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

AltoTrombone.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

AltoTrombone.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

AltoTrombone.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

AltoTrombone.all_clefs

Inherited from `instrumenttools._Instrument`

AltoTrombone.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`AltoTrombone.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`AltoTrombone.pitch_range`

Inherited from `instrumenttools._Instrument`

`AltoTrombone.primary_clefs`

Inherited from `instrumenttools._Instrument`

`AltoTrombone.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`AltoTrombone.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

AltoTrombone.**sounding_pitch_of_fingered_middle_c**

Inherited from `instrumenttools._Instrument`

Methods

AltoTrombone.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

AltoTrombone.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

AltoTrombone.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

AltoTrombone.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

AltoTrombone.**__call__**(*args)

Inherited from `marktools.Mark`

AltoTrombone.**__delattr__**(*args)

Inherited from `marktools.Mark`

AltoTrombone.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

AltoTrombone.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

AltoTrombone.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

```

AltoTrombone.__hash__()
    Inherited from contexttools.InstrumentMark

AltoTrombone.__le__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject

AltoTrombone.__lt__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

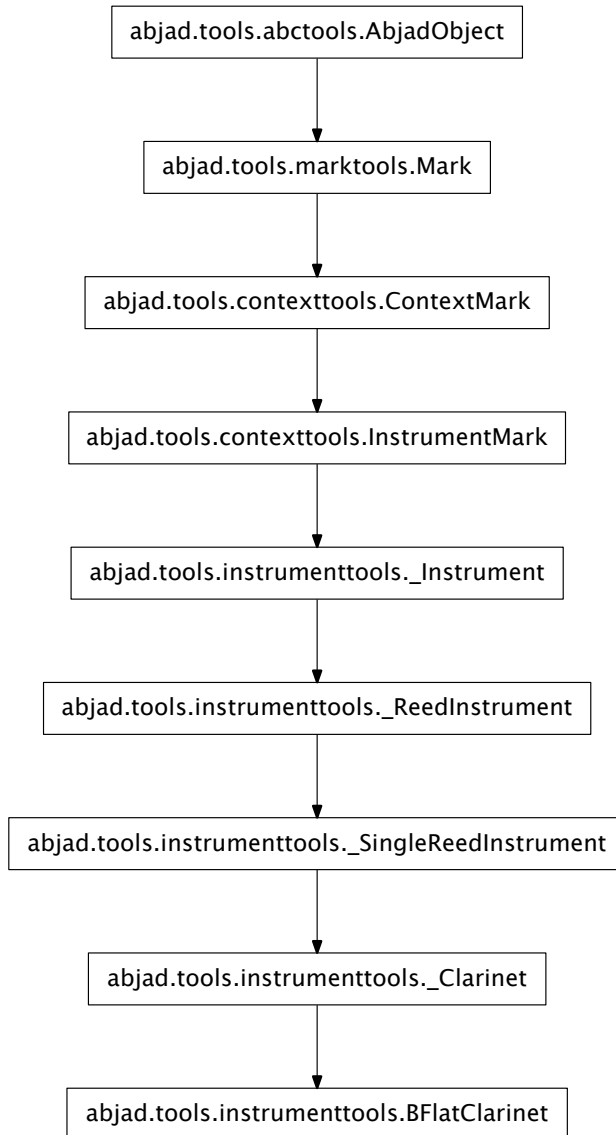
    Inherited from abctools.AbjadObject

AltoTrombone.__ne__(arg)
    Inherited from marktools.Mark

AltoTrombone.__repr__()
    Inherited from marktools.Mark

```

instrumenttools.BFlatClarinet



```

class instrumenttools.BFlatClarinet (**kwargs)
    New in version 2.0. Abjad model of the B-flat clarinet:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.BFlatClarinet() (staff)
    BFlatClarinet() (Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  c'8
  d'8
  e'8
  f'8
}
```

The B-flat clarinet targets staff context by default.

Read-only Properties

BFlatClarinet.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BFlatClarinet.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BFlatClarinet.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BFlatClarinet.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

BFlatClarinet.is_primary_instrument

Inherited from `instrumenttools._Instrument`

BFlatClarinet.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

BFlatClarinet.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

BFlatClarinet.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

BFlatClarinet.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

BFlatClarinet.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

BFlatClarinet.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BFlatClarinet.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

BFlatClarinet.all_clefs

Inherited from `instrumenttools._Instrument`

BFlatClarinet.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:


```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`BFlatClarinet.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`BFlatClarinet.pitch_range`

Inherited from `instrumenttools._Instrument`

`BFlatClarinet.primary_clefs`

Inherited from `instrumenttools._Instrument`

`BFlatClarinet.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`BFlatClarinet.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`BFlatClarinet.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`BFlatClarinet.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`BFlatClarinet.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`BFlatClarinet.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`BFlatClarinet.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`BFlatClarinet.__call__(*args)`

Inherited from `marktools.Mark`

`BFlatClarinet.__delattr__(*args)`

Inherited from `marktools.Mark`

`BFlatClarinet.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`BFlatClarinet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BFlatClarinet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

```
BFlatClarinet.__hash__()
    Inherited from contexttools.InstrumentMark

BFlatClarinet.__le__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject

BFlatClarinet.__lt__(arg)
    Abjad objects by default do not implement this method.

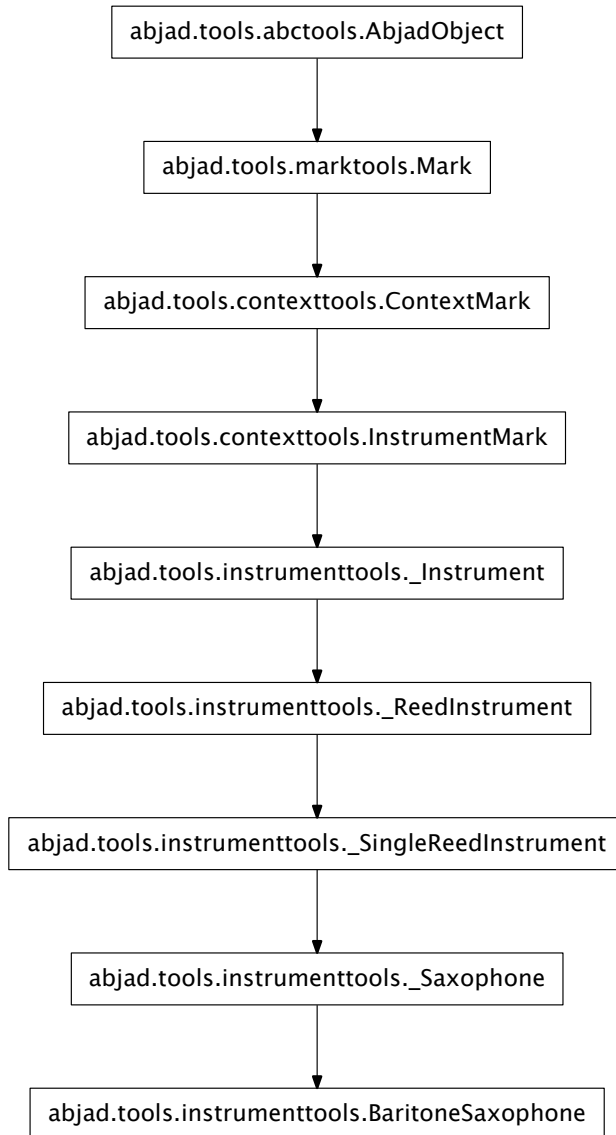
    Raise exception.

    Inherited from abctools.AbjadObject

BFlatClarinet.__ne__(arg)
    Inherited from marktools.Mark

BFlatClarinet.__repr__()
    Inherited from marktools.Mark
```

instrumenttools.BaritoneSaxophone



class instrumenttools.**BaritoneSaxophone** (**kwargs)

New in version 2.6. Abjad model of the baritone saxophone:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> instrumenttools.BaritoneSaxophone()(staff)
BaritoneSaxophone()(Staff{4})
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Baritone saxophone }
  \set Staff.shortInstrumentName = \markup { Bar. sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The baritone saxophone is pitched in E-flat.

The baritone saxophone targets staff context by default.

Read-only Properties

`BaritoneSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`BaritoneSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.is_primary_instrument`

Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.is_secondary_instrument`

Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.lilypond_format`

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`BaritoneSaxophone.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`BaritoneSaxophone.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`BaritoneSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

`BaritoneSaxophone.all_clefs`

Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.instrument_name`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BaritoneSaxophone.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

BaritoneSaxophone.pitch_range

Inherited from `instrumenttools._Instrument`

BaritoneSaxophone.primary_clefs

Inherited from `instrumenttools._Instrument`

BaritoneSaxophone.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BaritoneSaxophone.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`BaritoneSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`BaritoneSaxophone.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`BaritoneSaxophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`BaritoneSaxophone.__call__(*args)`

Inherited from `marktools.Mark`

`BaritoneSaxophone.__delattr__(*args)`

Inherited from `marktools.Mark`

`BaritoneSaxophone.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BaritoneSaxophone.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`BaritoneSaxophone.__hash__()`
Inherited from `contexttools.InstrumentMark`

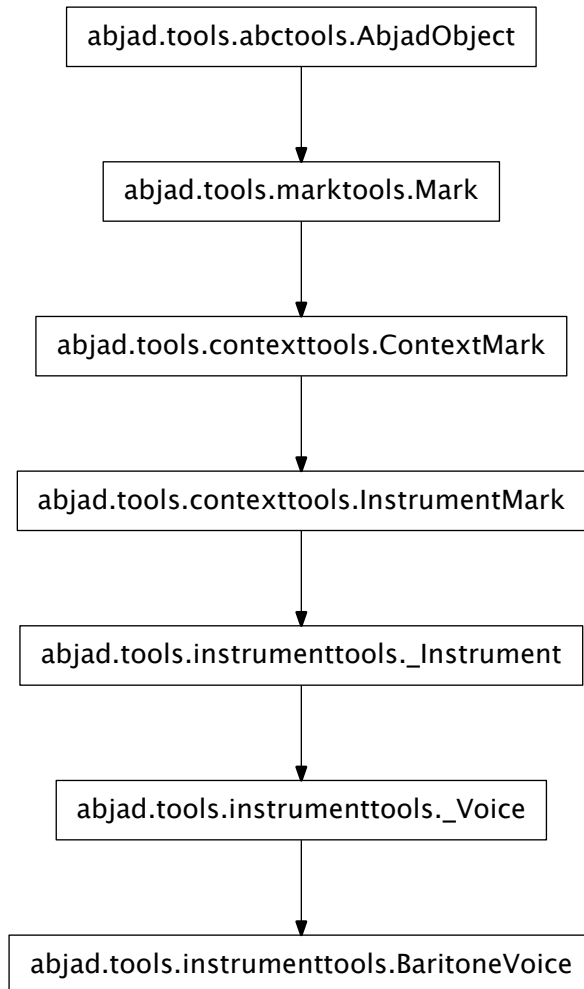
`BaritoneSaxophone.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`BaritoneSaxophone.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`BaritoneSaxophone.__ne__(arg)`
Inherited from `marktools.Mark`

`BaritoneSaxophone.__repr__()`
Inherited from `marktools.Mark`

instrumenttools.BaritoneVoice



```

class instrumenttools.BaritoneVoice(**kwargs)
    New in version 2.8. Abjad model of the baritone voice:

    >>> staff = Staff("c8 d8 e8 f8")

    >>> instrumenttools.BaritoneVoice()(staff)
    BaritoneVoice()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Baritone voice }
      \set Staff.shortInstrumentName = \markup { Baritone }
      c8
      d8
  
```

```

    e8
    f8
}

```

The baritone voice targets staff context by default.

Read-only Properties

BaritoneVoice.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BaritoneVoice.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BaritoneVoice.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BaritoneVoice.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

BaritoneVoice.is_primary_instrument

Inherited from `instrumenttools._Instrument`

BaritoneVoice.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

BaritoneVoice.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

BaritoneVoice.lilypond_format

Read-only LilyPond input format of instrument mark:

```

>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Inherited from `contexttools.InstrumentMark`

BaritoneVoice.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

BaritoneVoice.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

BaritoneVoice.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BaritoneVoice.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

BaritoneVoice.all_clefs

Inherited from `instrumenttools._Instrument`

BaritoneVoice.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BaritoneVoice.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

BaritoneVoice.pitch_range

Inherited from `instrumenttools._Instrument`

BaritoneVoice.primary_clefs

Inherited from `instrumenttools._Instrument`

BaritoneVoice.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BaritoneVoice.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

BaritoneVoice.sounding_pitch_of_fingered_middle_c

Inherited from `instrumenttools._Instrument`

Methods

BaritoneVoice.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from contexttools.ContextMark

BaritoneVoice.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from contexttools.ContextMark

BaritoneVoice.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from instrumenttools._Instrument

BaritoneVoice.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from instrumenttools._Instrument

Special Methods

BaritoneVoice.**__call__**(*args)

Inherited from marktools.Mark

BaritoneVoice.**__delattr__**(*args)

Inherited from marktools.Mark

BaritoneVoice.**__eq__**(arg)

Inherited from contexttools.InstrumentMark

BaritoneVoice.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

BaritoneVoice.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

BaritoneVoice.**__hash__**()

Inherited from contexttools.InstrumentMark

BaritoneVoice.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

BaritoneVoice.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

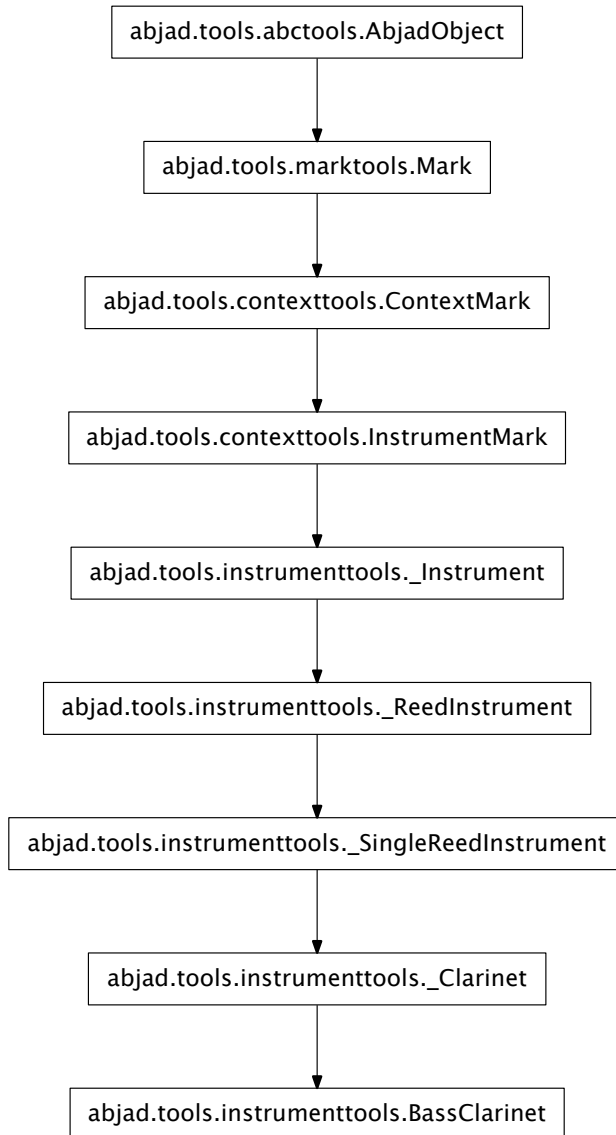
BaritoneVoice.__ne__(arg)

Inherited from `marktools.Mark`

BaritoneVoice.__repr__()

Inherited from `marktools.Mark`

instrumenttools.BassClarinet



class instrumenttools.**BassClarinet** (**kwargs)

New in version 2.0. Abjad model of the bass clarinet:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> instrumenttools.BassClarinet()(staff)
BassClarinet()(Staff{4})
```



```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Bass clarinet }
  \set Staff.shortInstrumentName = \markup { Bass cl. }
  c'8
  d'8
  e'8
  f'8
}
```

The bass clarinet targets staff context by default.

Read-only Properties

BassClarinet.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassClarinet.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassClarinet.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassClarinet.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

BassClarinet.is_primary_instrument

Inherited from `instrumenttools._Instrument`

BassClarinet.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

BassClarinet.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

BassClarinet.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

BassClarinet.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

BassClarinet.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

BassClarinet.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassClarinet.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

BassClarinet.all_clefs

Inherited from `instrumenttools._Instrument`

BassClarinet.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`BassClarinet.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`BassClarinet.pitch_range`

Inherited from `instrumenttools._Instrument`

`BassClarinet.primary_clefs`

Inherited from `instrumenttools._Instrument`

`BassClarinet.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`BassClarinet.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`BassClarinet.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`BassClarinet.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`BassClarinet.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`BassClarinet.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`BassClarinet.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`BassClarinet.__call__(*args)`

Inherited from `marktools.Mark`

`BassClarinet.__delattr__(*args)`

Inherited from `marktools.Mark`

`BassClarinet.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`BassClarinet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BassClarinet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

```
BassClarinet.__hash__()
    Inherited from contexttools.InstrumentMark

BassClarinet.__le__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject

BassClarinet.__lt__(arg)
    Abjad objects by default do not implement this method.

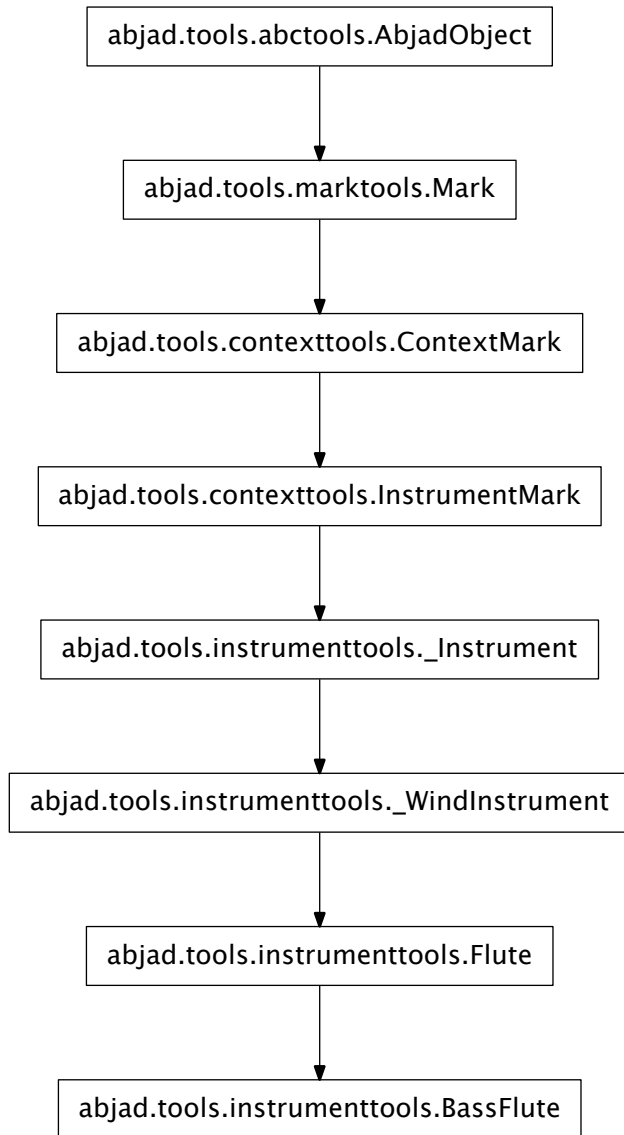
    Raise exception.

    Inherited from abctools.AbjadObject

BassClarinet.__ne__(arg)
    Inherited from marktools.Mark

BassClarinet.__repr__()
    Inherited from marktools.Mark
```

instrumenttools.BassFlute



```

class instrumenttools.BassFlute(**kwargs)
    New in version 2.0. Abjad model of the bass flute:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.BassFlute()(staff)
    BassFlute()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Bass flute }
  \set Staff.shortInstrumentName = \markup { Bass fl. }
  c'8
  d'8
  e'8
  f'8
}
```

The bass flute targets staff context by default.

Read-only Properties

BassFlute.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassFlute.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassFlute.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassFlute.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

BassFlute.is_primary_instrument

Inherited from `instrumenttools._Instrument`

BassFlute.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

BassFlute.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

BassFlute.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

BassFlute.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

BassFlute.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

BassFlute.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassFlute.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

BassFlute.all_clefs

Inherited from `instrumenttools._Instrument`

BassFlute.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:


```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BassFlute.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

BassFlute.pitch_range

Inherited from `instrumenttools._Instrument`

BassFlute.primary_clefs

Inherited from `instrumenttools._Instrument`

BassFlute.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BassFlute.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`BassFlute.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`BassFlute.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`BassFlute.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`BassFlute.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`BassFlute.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`BassFlute.__call__(*args)`

Inherited from `marktools.Mark`

`BassFlute.__delattr__(*args)`

Inherited from `marktools.Mark`

`BassFlute.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`BassFlute.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BassFlute.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BassFlute.__hash__()`
Inherited from `contexttools.InstrumentMark`

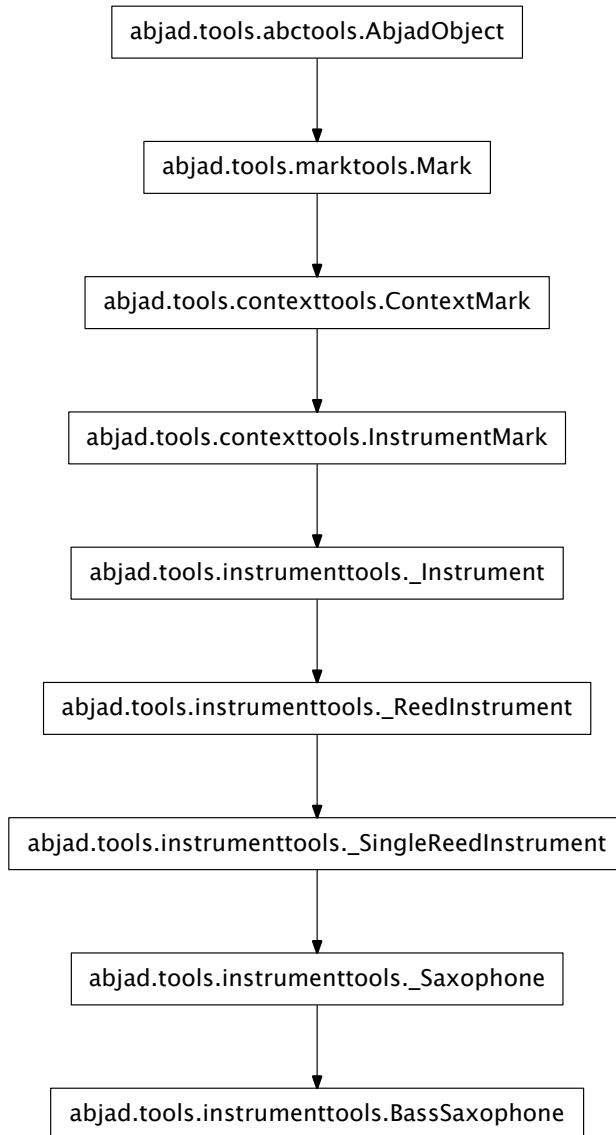
`BassFlute.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`BassFlute.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`BassFlute.__ne__(arg)`
Inherited from `marktools.Mark`

`BassFlute.__repr__()`
Inherited from `marktools.Mark`

instrumenttools.BassSaxophone



```

class instrumenttools.BassSaxophone(**kwargs)
    New in version 2.6. Abjad model of the bass saxophone:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.BassSaxophone()(staff)
    BassSaxophone()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Bass saxophone }
  \set Staff.shortInstrumentName = \markup { Bass sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The bass saxophone is pitched in B-flat.

The bass saxophone targets staff context by default.

Read-only Properties

BassSaxophone.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassSaxophone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassSaxophone.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassSaxophone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

BassSaxophone.is_primary_instrument

Inherited from `instrumenttools._Instrument`

BassSaxophone.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

BassSaxophone.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

BassSaxophone.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

BassSaxophone.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

BassSaxophone.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

BassSaxophone.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassSaxophone.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

BassSaxophone.all_clefs

Inherited from `instrumenttools._Instrument`

BassSaxophone.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BassSaxophone.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

BassSaxophone.pitch_range

Inherited from `instrumenttools._Instrument`

BassSaxophone.primary_clefs

Inherited from `instrumenttools._Instrument`

BassSaxophone.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BassSaxophone.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`BassSaxophone.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`BassSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`BassSaxophone.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`BassSaxophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`BassSaxophone.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`BassSaxophone.__call__(*args)`

Inherited from `marktools.Mark`

`BassSaxophone.__delattr__(*args)`

Inherited from `marktools.Mark`

`BassSaxophone.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`BassSaxophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BassSaxophone.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BassSaxophone.__hash__()`

Inherited from `contexttools.InstrumentMark`

`BassSaxophone.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BassSaxophone.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

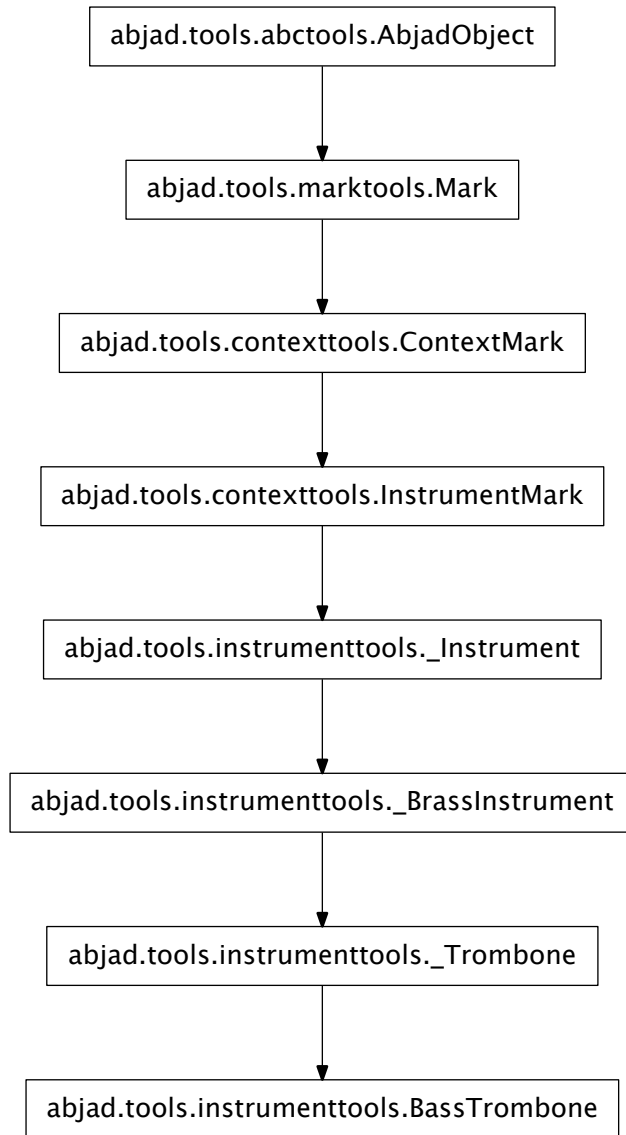
`BassSaxophone.__ne__(arg)`

Inherited from `marktools.Mark`

`BassSaxophone.__repr__()`

Inherited from `marktools.Mark`

instrumenttools.BassTrombone



```

class instrumenttools.BassTrombone(**kwargs)
    New in version 2.0. Abjad model of the tenor trombone:

    >>> staff = Staff("c'8 d'8 e'8 f'8")
    >>> contexttools.ClefMark('bass')(staff)
    ClefMark('bass')(Staff{4})

    >>> instrumenttools.BassTrombone()(staff)
    BassTrombone()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Bass trombone }
  \set Staff.shortInstrumentName = \markup { Bass trb. }
  c'8
  d'8
  e'8
  f'8
}
```

The tenor trombone targets staff context by default.

Read-only Properties

BassTrombone.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassTrombone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassTrombone.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassTrombone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

BassTrombone.is_primary_instrument

Inherited from `instrumenttools._Instrument`

BassTrombone.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

BassTrombone.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

BassTrombone.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

BassTrombone.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

BassTrombone.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

BassTrombone.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassTrombone.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

BassTrombone.all_clefs

Inherited from `instrumenttools._Instrument`

BassTrombone.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BassTrombone.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

BassTrombone.pitch_range

Inherited from `instrumenttools._Instrument`

BassTrombone.primary_clefs

Inherited from `instrumenttools._Instrument`

BassTrombone.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BassTrombone.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

BassTrombone.**sounding_pitch_of_fingered_middle_c**

Inherited from `instrumenttools._Instrument`

Methods

BassTrombone.**attach** (*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

BassTrombone.**detach** ()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

BassTrombone.**get_default_performer_name** (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

BassTrombone.**get_performer_names** ()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

BassTrombone.**__call__** (*args)

Inherited from `marktools.Mark`

BassTrombone.**__delattr__** (*args)

Inherited from `marktools.Mark`

BassTrombone.**__eq__** (arg)

Inherited from `contexttools.InstrumentMark`

BassTrombone.**__ge__** (arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

BassTrombone.**__gt__** (arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

```

BassTrombone.__hash__()
    Inherited from contexttools.InstrumentMark

BassTrombone.__le__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject

BassTrombone.__lt__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

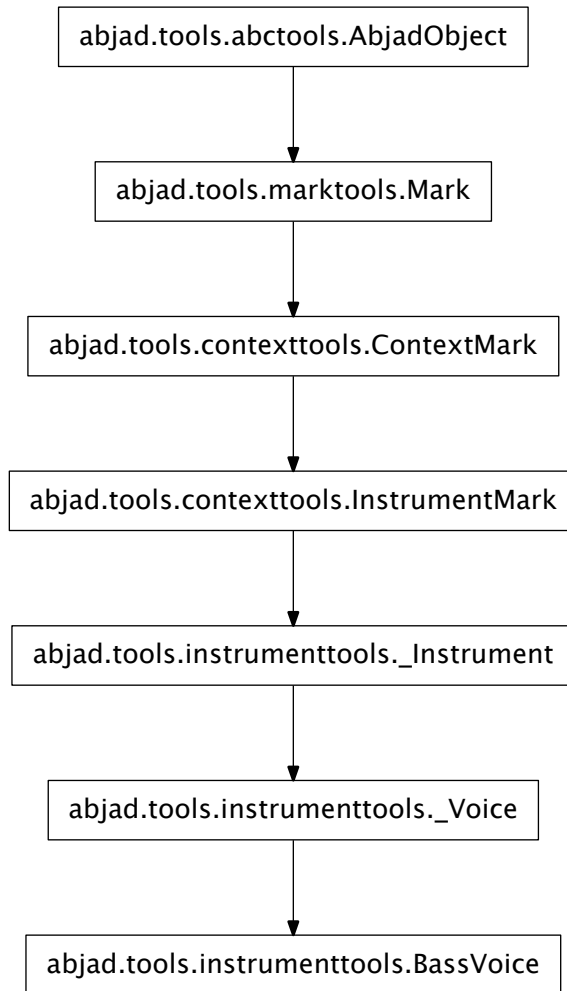
    Inherited from abctools.AbjadObject

BassTrombone.__ne__(arg)
    Inherited from marktools.Mark

BassTrombone.__repr__()
    Inherited from marktools.Mark

```

instrumenttools.BassVoice



```

class instrumenttools.BassVoice(**kwargs)
    New in version 2.8. Abjad model of the bass voice:

    >>> staff = Staff("c8 d8 e8 f8")

    >>> instrumenttools.BassVoice()(staff)
    BassVoice()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Bass voice }
      \set Staff.shortInstrumentName = \markup { Bass }
      c8
      d8
  
```



```

        e8
        f8
    }

```

The bass voice targets staff context by default.

Read-only Properties

BassVoice.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassVoice.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

BassVoice.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassVoice.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

BassVoice.is_primary_instrument

Inherited from `instrumenttools._Instrument`

BassVoice.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

BassVoice.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

BassVoice.lilypond_format

Read-only LilyPond input format of instrument mark:

```

>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Inherited from `contexttools.InstrumentMark`

BassVoice.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

BassVoice.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

BassVoice.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

BassVoice.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

BassVoice.all_clefs

Inherited from `instrumenttools._Instrument`

BassVoice.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

BassVoice.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

BassVoice.pitch_range

Inherited from instrumenttools._Instrument

BassVoice.primary_clefs

Inherited from instrumenttools._Instrument

BassVoice.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

BassVoice.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

BassVoice.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

`BassVoice.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`BassVoice.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`BassVoice.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`BassVoice.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`BassVoice.__call__(*args)`

Inherited from `marktools.Mark`

`BassVoice.__delattr__(*args)`

Inherited from `marktools.Mark`

`BassVoice.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`BassVoice.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BassVoice.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BassVoice.__hash__()`

Inherited from `contexttools.InstrumentMark`

BassVoice.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

BassVoice.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

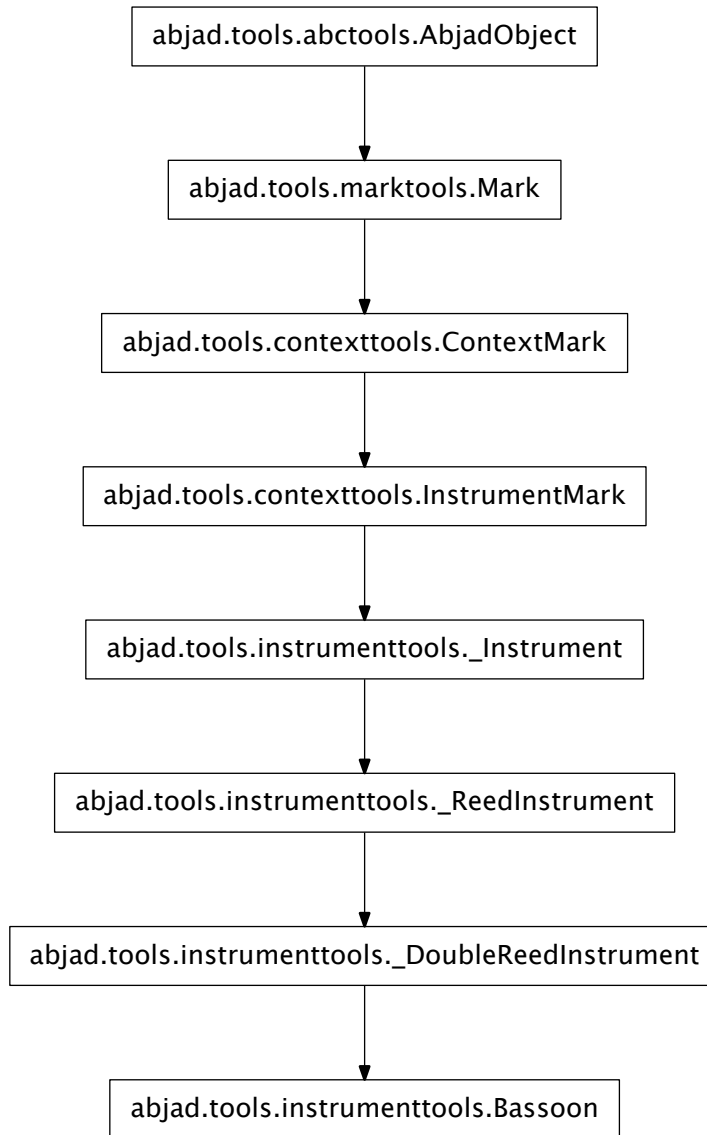
BassVoice.**__ne__**(arg)

Inherited from `marktools.Mark`

BassVoice.**__repr__**()

Inherited from `marktools.Mark`

instrumenttools.Bassoon



```

class instrumenttools.Bassoon(**kwargs)
    New in version 2.0. Abjad model of the bassoon:

    >>> staff = Staff("c'8 d'8 e'8 f'8")
    >>> contexttools.ClefMark('bass')(staff)
    ClefMark('bass')(Staff{4})

    >>> instrumenttools.Bassoon()(staff)
    Bassoon()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Bassoon }
  \set Staff.shortInstrumentName = \markup { Bsn. }
  c'8
  d'8
  e'8
  f'8
}
```

The bassoon targets staff context by default.

Read-only Properties

Bassoon.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Bassoon.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Bassoon.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Bassoon.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Bassoon.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Bassoon.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Bassoon.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Bassoon.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Bassoon.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Bassoon.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Bassoon.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Bassoon.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Bassoon.all_clefs

Inherited from `instrumenttools._Instrument`

Bassoon.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:


```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Bassoon.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Bassoon.pitch_range

Inherited from `instrumenttools._Instrument`

Bassoon.primary_clefs

Inherited from `instrumenttools._Instrument`

Bassoon.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Bassoon.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Bassoon.**sounding_pitch_of_fingered_middle_c**

Inherited from `instrumenttools._Instrument`

Methods

Bassoon.**attach** (*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Bassoon.**detach** ()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Bassoon.**get_default_performer_name** (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Bassoon.**get_performer_names** ()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Bassoon.**__call__** (*args)

Inherited from `marktools.Mark`

Bassoon.**__delattr__** (*args)

Inherited from `marktools.Mark`

Bassoon.**__eq__** (arg)

Inherited from `contexttools.InstrumentMark`

Bassoon.**__ge__** (arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Bassoon.**__gt__** (arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Bassoon.__hash__()`
 Inherited from `contexttools.InstrumentMark`

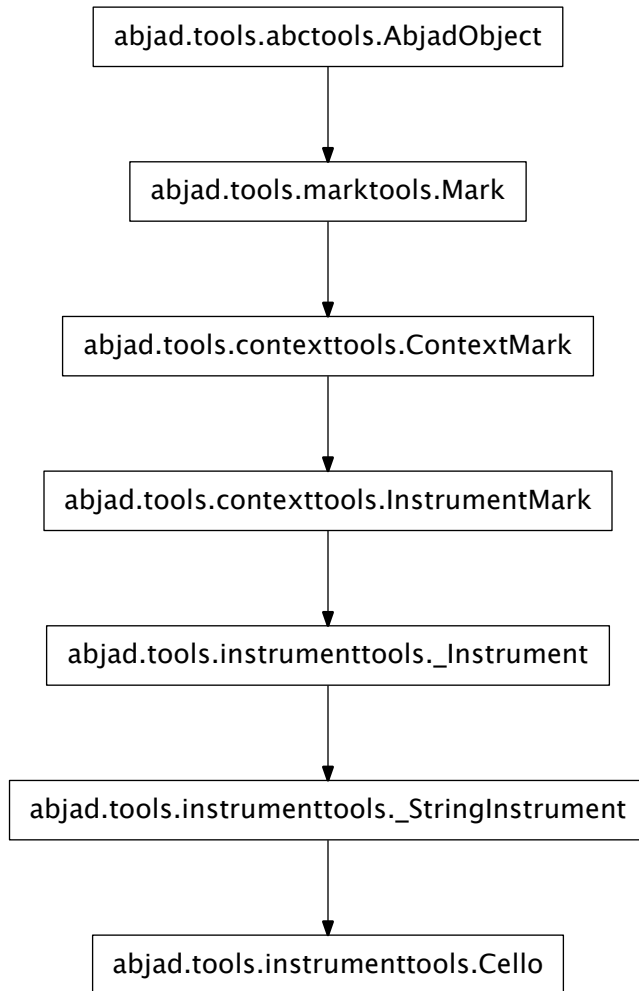
`Bassoon.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Bassoon.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Bassoon.__ne__(arg)`
 Inherited from `marktools.Mark`

`Bassoon.__repr__()`
 Inherited from `marktools.Mark`

instrumenttools.Cello



class instrumenttools.**Cello** (**kwargs)

New in version 2.0. Abjad model of the cello:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

>>> instrumenttools.Cello()(staff)
Cello()(Staff{4})

>>> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Cello }

```

```

\set Staff.shortInstrumentName = \markup { Vc. }
c' 8
d' 8
e' 8
f' 8
}

```

The cello targets staff context by default.

Read-only Properties

Cello.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Cello.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Cello.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c' 4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Cello.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Cello.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Cello.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Cello.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Cello.lilypond_format

Read-only LilyPond input format of instrument mark:

```

>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Inherited from `contexttools.InstrumentMark`

Cello.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Cello.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Cello.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Cello.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Cello.all_clefs

Inherited from `instrumenttools._Instrument`

Cello.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Cello.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Cello.pitch_range

Inherited from instrumenttools._Instrument

Cello.primary_clefs

Inherited from instrumenttools._Instrument

Cello.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Cello.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Cello.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

`Cello.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`Cello.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`Cello.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`Cello.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`Cello.__call__(*args)`

Inherited from `marktools.Mark`

`Cello.__delattr__(*args)`

Inherited from `marktools.Mark`

`Cello.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`Cello.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Cello.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Cello.__hash__()`

Inherited from `contexttools.InstrumentMark`

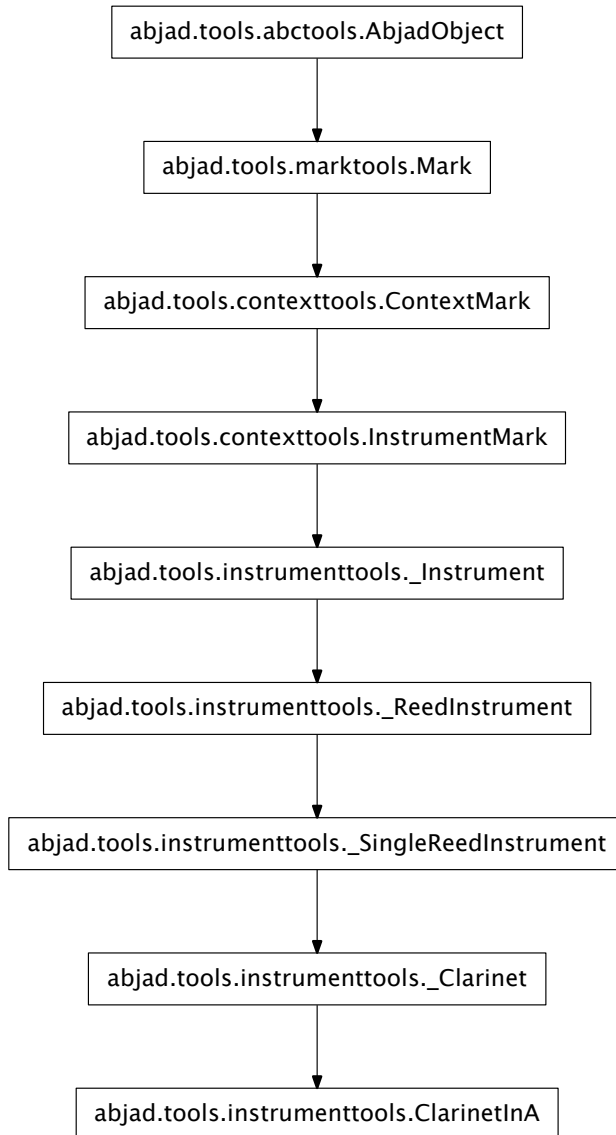
`Cello.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`Cello.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`Cello.__ne__(arg)`
Inherited from `marktools.Mark`

`Cello.__repr__()`
Inherited from `marktools.Mark`

instrumenttools.ClarinetInA



```

class instrumenttools.ClarinetInA(**kwargs)
    New in version 2.6. Abjad model of the clarinet in A:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.ClarinetInA()(staff)
    ClarinetInA()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in A }
  \set Staff.shortInstrumentName = \markup { Cl. A \natural }
  c'8
  d'8
  e'8
  f'8
}
```

The clarinet in A targets staff context by default.

Read-only Properties

ClarinetInA.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

ClarinetInA.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

ClarinetInA.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

ClarinetInA.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

ClarinetInA.is_primary_instrument

Inherited from `instrumenttools._Instrument`

ClarinetInA.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

ClarinetInA.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

ClarinetInA.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

ClarinetInA.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

ClarinetInA.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

ClarinetInA.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

ClarinetInA.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

ClarinetInA.all_clefs

Inherited from `instrumenttools._Instrument`

ClarinetInA.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`ClarinetInA.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`ClarinetInA.pitch_range`

Inherited from `instrumenttools._Instrument`

`ClarinetInA.primary_clefs`

Inherited from `instrumenttools._Instrument`

`ClarinetInA.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`ClarinetInA.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

ClarinetInA.**sounding_pitch_of_fingered_middle_c**

Inherited from `instrumenttools._Instrument`

Methods

ClarinetInA.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

ClarinetInA.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

ClarinetInA.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

ClarinetInA.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

ClarinetInA.**__call__**(*args)

Inherited from `marktools.Mark`

ClarinetInA.**__delattr__**(*args)

Inherited from `marktools.Mark`

ClarinetInA.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

ClarinetInA.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

ClarinetInA.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

ClarinetInA.__hash__()
Inherited from contexttools.InstrumentMark

ClarinetInA.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Inherited from abctools.AbjadObject

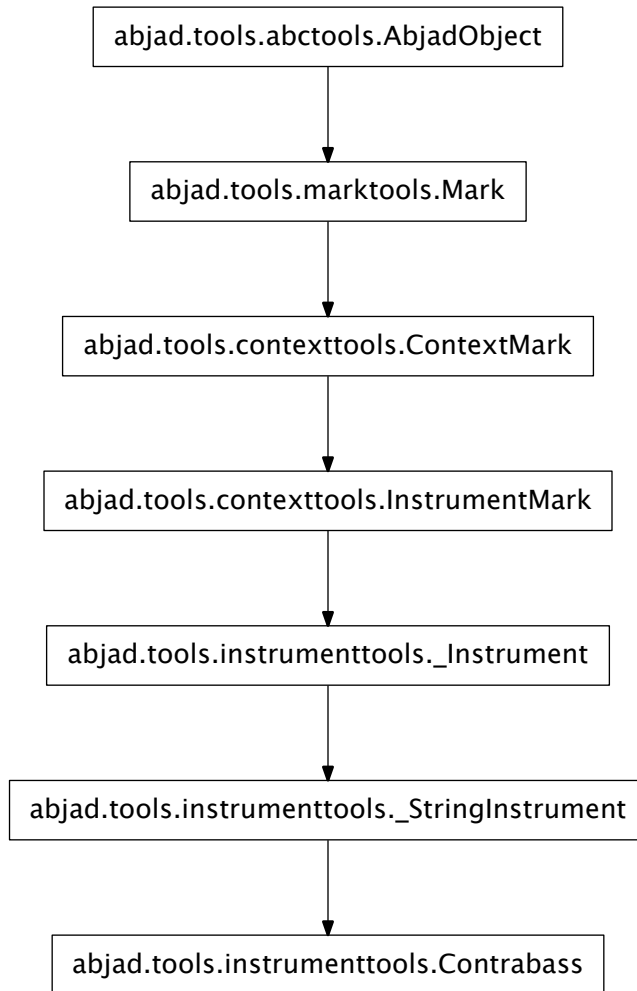
ClarinetInA.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Inherited from abctools.AbjadObject

ClarinetInA.__ne__(arg)
Inherited from marktools.Mark

ClarinetInA.__repr__()
Inherited from marktools.Mark

instrumenttools.Contrabass



```

class instrumenttools.Contrabass(**kwargs)
    New in version 2.0. Abjad model of the contrabass:

    >>> staff = Staff("c'8 d'8 e'8 f'8")
    >>> contexttools.ClefMark('bass')(staff)
    ClefMark('bass')(Staff{4})

    >>> instrumenttools.Contrabass()(staff)
    Contrabass()(Staff{4})

    >>> f(staff)
    \new Staff {
      \clef "bass"
      \set Staff.instrumentName = \markup { Contrabass }
  
```



```
\set Staff.shortInstrumentName = \markup { Vb. }
c' 8
d' 8
e' 8
f' 8
}
```

The contrabass targets staff context by default.

Read-only Properties

Contrabass.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Contrabass.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Contrabass.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Contrabass.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Contrabass.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Contrabass.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Contrabass.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Contrabass.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Contrabass.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Contrabass.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Contrabass.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Contrabass.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Contrabass.all_clefs

Inherited from `instrumenttools._Instrument`

Contrabass.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Contrabass.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Contrabass.pitch_range

Inherited from instrumenttools._Instrument

Contrabass.primary_clefs

Inherited from instrumenttools._Instrument

Contrabass.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Contrabass.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Contrabass.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

`Contrabass.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`Contrabass.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`Contrabass.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`Contrabass.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`Contrabass.__call__(*args)`

Inherited from `marktools.Mark`

`Contrabass.__delattr__(*args)`

Inherited from `marktools.Mark`

`Contrabass.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`Contrabass.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Contrabass.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Contrabass.__hash__()`

Inherited from `contexttools.InstrumentMark`

`Contrabass.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Contrabass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

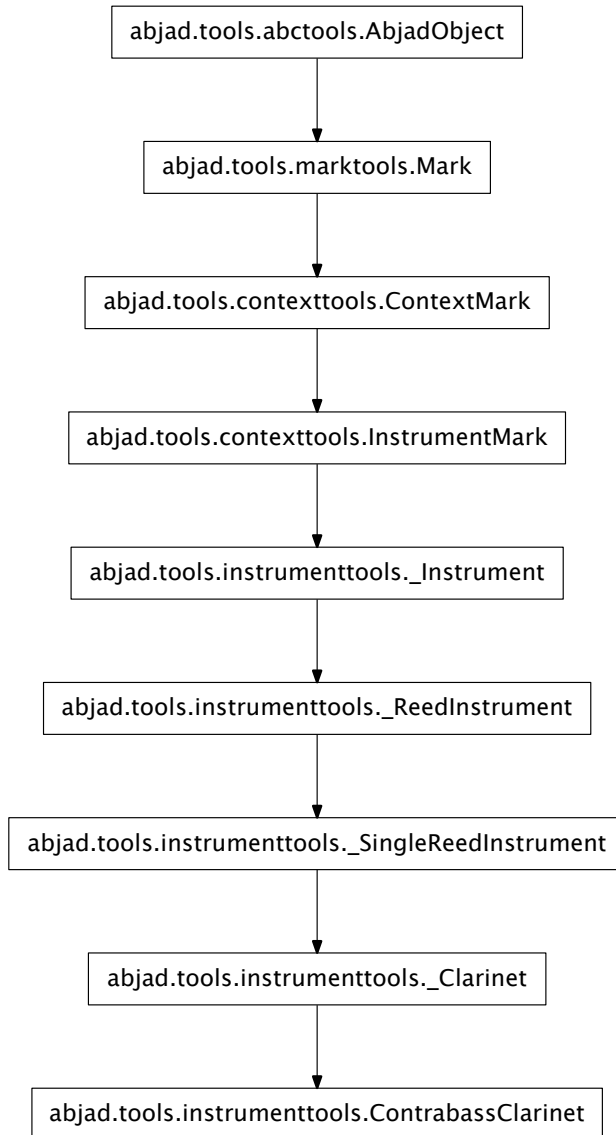
`Contrabass.__ne__(arg)`

Inherited from `marktools.Mark`

`Contrabass.__repr__()`

Inherited from `marktools.Mark`

instrumenttools.ContrabassClarinet



class instrumenttools.**ContrabassClarinet** (**kwargs)

New in version 2.6. Abjad model of the contrassbass clarinet:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> instrumenttools.ContrabassClarinet()(staff)
ContrabassClarinet()(Staff{4})
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Contrabass clarinet }
  \set Staff.shortInstrumentName = \markup { Cbass cl. }
  c'8
  d'8
  e'8
  f'8
}
```

The contrabass clarinet targets staff context by default.

Read-only Properties

`ContrabassClarinet.default_instrument_name`

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.effective_context`

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`ContrabassClarinet.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.is_primary_instrument`

Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.is_secondary_instrument`

Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.lilypond_format`

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`ContrabassClarinet.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`ContrabassClarinet.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`ContrabassClarinet.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

`ContrabassClarinet.all_clefs`

Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.instrument_name`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:


```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.pitch_range`

Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.primary_clefs`

Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`ContrabassClarinet.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`ContrabassClarinet.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`ContrabassClarinet.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`ContrabassClarinet.__call__(*args)`

Inherited from `marktools.Mark`

`ContrabassClarinet.__delattr__(*args)`

Inherited from `marktools.Mark`

`ContrabassClarinet.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ContrabassClarinet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ContrabassClarinet.__hash__()`
Inherited from `contexttools.InstrumentMark`

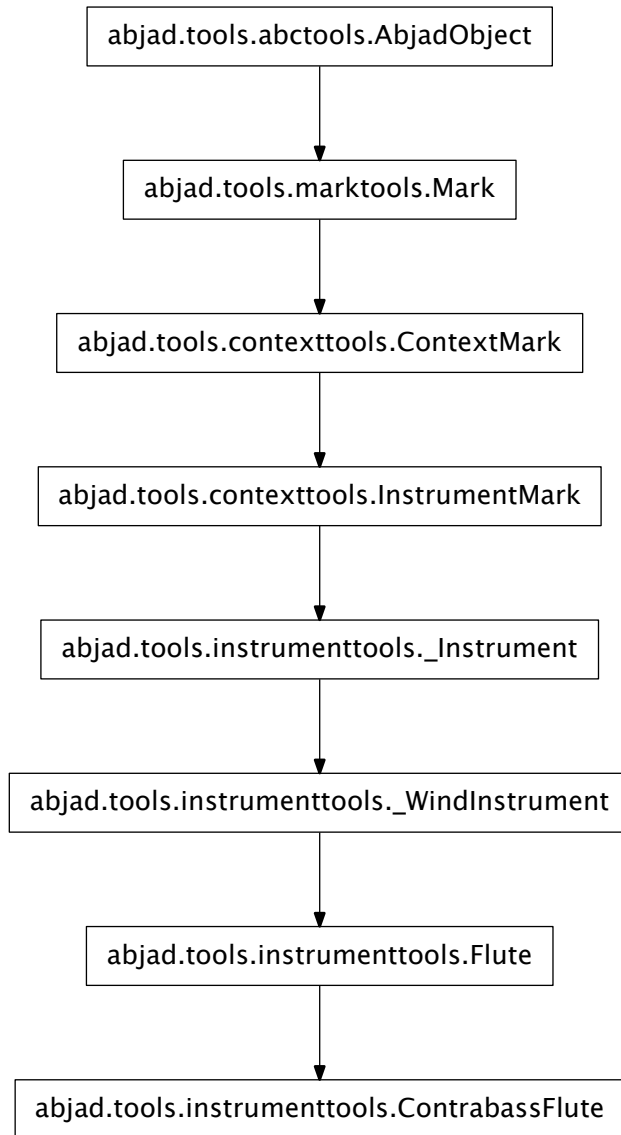
`ContrabassClarinet.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`ContrabassClarinet.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`ContrabassClarinet.__ne__(arg)`
Inherited from `marktools.Mark`

`ContrabassClarinet.__repr__()`
Inherited from `marktools.Mark`

instrumenttools.ContrabassFlute



```

class instrumenttools.ContrabassFlute (**kwargs)
    New in version 2.0. Abjad model of the contrabass flute:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.ContrabassFlute() (staff)
    ContrabassFlute() (Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Contrabass flute }
  \set Staff.shortInstrumentName = \markup { Cbass. fl. }
  c'8
  d'8
  e'8
  f'8
}
```

The contrabass flute targets staff context by default.

Read-only Properties

ContrabassFlute.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

ContrabassFlute.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

ContrabassFlute.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

ContrabassFlute.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

ContrabassFlute.is_primary_instrument

Inherited from `instrumenttools._Instrument`

ContrabassFlute.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

ContrabassFlute.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

ContrabassFlute.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`ContrabassFlute.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`ContrabassFlute.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`ContrabassFlute.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

`ContrabassFlute.all_clefs`

Inherited from `instrumenttools._Instrument`

`ContrabassFlute.instrument_name`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.pitch_range`

Inherited from `instrumenttools._Instrument`

`ContrabassFlute.primary_clefs`

Inherited from `instrumenttools._Instrument`

`ContrabassFlute.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`ContrabassFlute.attach` (*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`ContrabassFlute.detach` ()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`ContrabassFlute.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`ContrabassFlute.get_performer_names` ()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`ContrabassFlute.__call__` (*args)

Inherited from `marktools.Mark`

`ContrabassFlute.__delattr__` (*args)

Inherited from `marktools.Mark`

`ContrabassFlute.__eq__` (arg)

Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ContrabassFlute.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`


```

ContrabassFlute.__hash__()
    Inherited from contexttools.InstrumentMark

ContrabassFlute.__le__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject

ContrabassFlute.__lt__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

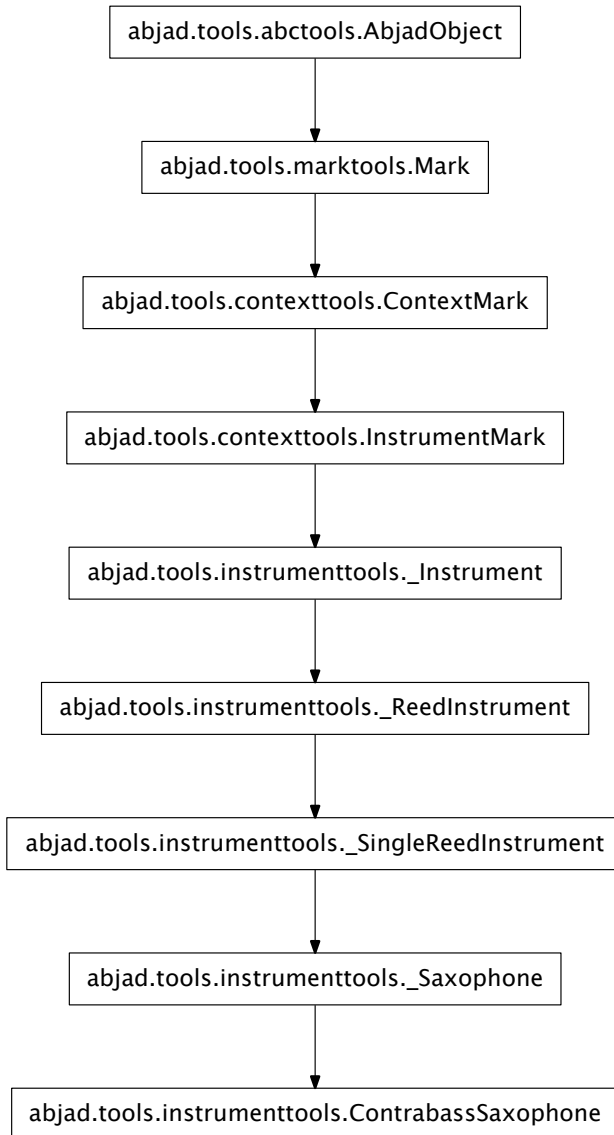
    Inherited from abctools.AbjadObject

ContrabassFlute.__ne__(arg)
    Inherited from marktools.Mark

ContrabassFlute.__repr__()
    Inherited from marktools.Mark

```

instrumenttools.ContrabassSaxophone



class instrumenttools.**ContrabassSaxophone** (**kwargs)

New in version 2.6. Abjad model of the bass saxophone:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> instrumenttools.ContrabassSaxophone()(staff)
ContrabassSaxophone()(Staff{4})
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Contrabass saxophone }
  \set Staff.shortInstrumentName = \markup { Cbass. sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The contrabass saxophone is pitched in E-flat.

The contrabass saxophone targets staff context by default.

Read-only Properties

`ContrabassSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`ContrabassSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.is_primary_instrument`

Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.is_secondary_instrument`

Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

ContrabassSaxophone.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

ContrabassSaxophone.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

ContrabassSaxophone.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

ContrabassSaxophone.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

ContrabassSaxophone.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

ContrabassSaxophone.all_clefs

Inherited from `instrumenttools._Instrument`

ContrabassSaxophone.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.pitch_range`

Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.primary_clefs`

Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`ContrabassSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`ContrabassSaxophone.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`ContrabassSaxophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`ContrabassSaxophone.__call__(*args)`

Inherited from `marktools.Mark`

`ContrabassSaxophone.__delattr__(*args)`

Inherited from `marktools.Mark`

`ContrabassSaxophone.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ContrabassSaxophone.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`ContrabassSaxophone.__hash__()`
Inherited from `contexttools.InstrumentMark`

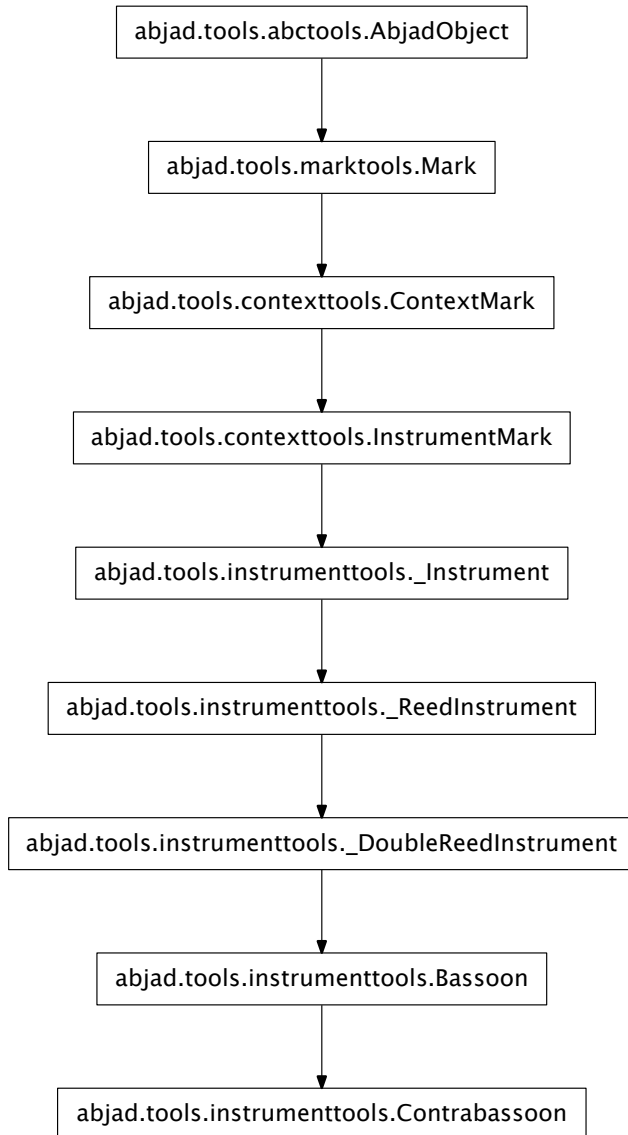
`ContrabassSaxophone.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`ContrabassSaxophone.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`ContrabassSaxophone.__ne__(arg)`
Inherited from `marktools.Mark`

`ContrabassSaxophone.__repr__()`
Inherited from `marktools.Mark`

instrumenttools.Contrabassoon



class instrumenttools.**Contrabassoon** (**kwargs)

New in version 2.0. Abjad model of the contrabassoon:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

>>> instrumenttools.Contrabassoon()(staff)
Contrabassoon()(Staff{4})
  
```



```
>>> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Contrabassoon }
  \set Staff.shortInstrumentName = \markup { Contrabsn. }
  c'8
  d'8
  e'8
  f'8
}
```

The contrabassoon targets staff context by default.

Read-only Properties

Contrabassoon.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Contrabassoon.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Contrabassoon.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Contrabassoon.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Contrabassoon.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Contrabassoon.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Contrabassoon.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Contrabassoon.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Contrabassoon.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Contrabassoon.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Contrabassoon.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Contrabassoon.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Contrabassoon.all_clefs

Inherited from `instrumenttools._Instrument`

Contrabassoon.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`Contrabassoon.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`Contrabassoon.pitch_range`

Inherited from `instrumenttools._Instrument`

`Contrabassoon.primary_clefs`

Inherited from `instrumenttools._Instrument`

`Contrabassoon.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`Contrabassoon.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`Contrabassoon.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`Contrabassoon.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`Contrabassoon.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`Contrabassoon.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`Contrabassoon.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`Contrabassoon.__call__(*args)`

Inherited from `marktools.Mark`

`Contrabassoon.__delattr__(*args)`

Inherited from `marktools.Mark`

`Contrabassoon.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`Contrabassoon.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Contrabassoon.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Contrabassoon.**__hash__**()
Inherited from contexttools.InstrumentMark

Contrabassoon.**__le__**(arg)
Abjad objects by default do not implement this method.
Raise exception.

Inherited from abctools.AbjadObject

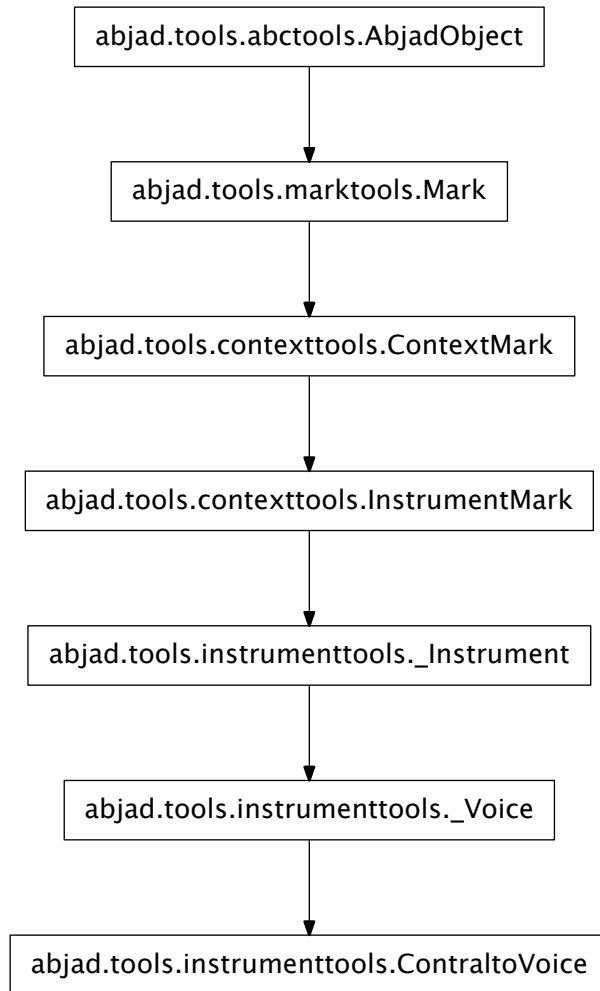
Contrabassoon.**__lt__**(arg)
Abjad objects by default do not implement this method.
Raise exception.

Inherited from abctools.AbjadObject

Contrabassoon.**__ne__**(arg)
Inherited from marktools.Mark

Contrabassoon.**__repr__**()
Inherited from marktools.Mark

instrumenttools.ContraltoVoice



```

class instrumenttools.ContraltoVoice(**kwargs)
    New in version 2.8. Abjad model of the contralto voice:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.ContraltoVoice()(staff)
    ContraltoVoice()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Contralto voice }
      \set Staff.shortInstrumentName = \markup { Contralto }
      c'8
      d'8
  
```

```
e' 8
f' 8
}
```

The contralto voice targets staff context by default.

Read-only Properties

ContraltoVoice.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

ContraltoVoice.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

ContraltoVoice.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

ContraltoVoice.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

ContraltoVoice.is_primary_instrument

Inherited from `instrumenttools._Instrument`

ContraltoVoice.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

ContraltoVoice.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

ContraltoVoice.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`ContraltoVoice.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`ContraltoVoice.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`ContraltoVoice.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

`ContraltoVoice.all_clefs`

Inherited from `instrumenttools._Instrument`

`ContraltoVoice.instrument_name`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

ContraltoVoice.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

ContraltoVoice.pitch_range

Inherited from instrumenttools._Instrument

ContraltoVoice.primary_clefs

Inherited from instrumenttools._Instrument

ContraltoVoice.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

ContraltoVoice.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

ContraltoVoice.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

`ContraltoVoice.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`ContraltoVoice.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`ContraltoVoice.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`ContraltoVoice.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`ContraltoVoice.__call__(*args)`

Inherited from `marktools.Mark`

`ContraltoVoice.__delattr__(*args)`

Inherited from `marktools.Mark`

`ContraltoVoice.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ContraltoVoice.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ContraltoVoice.__hash__()`

Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ContraltoVoice.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

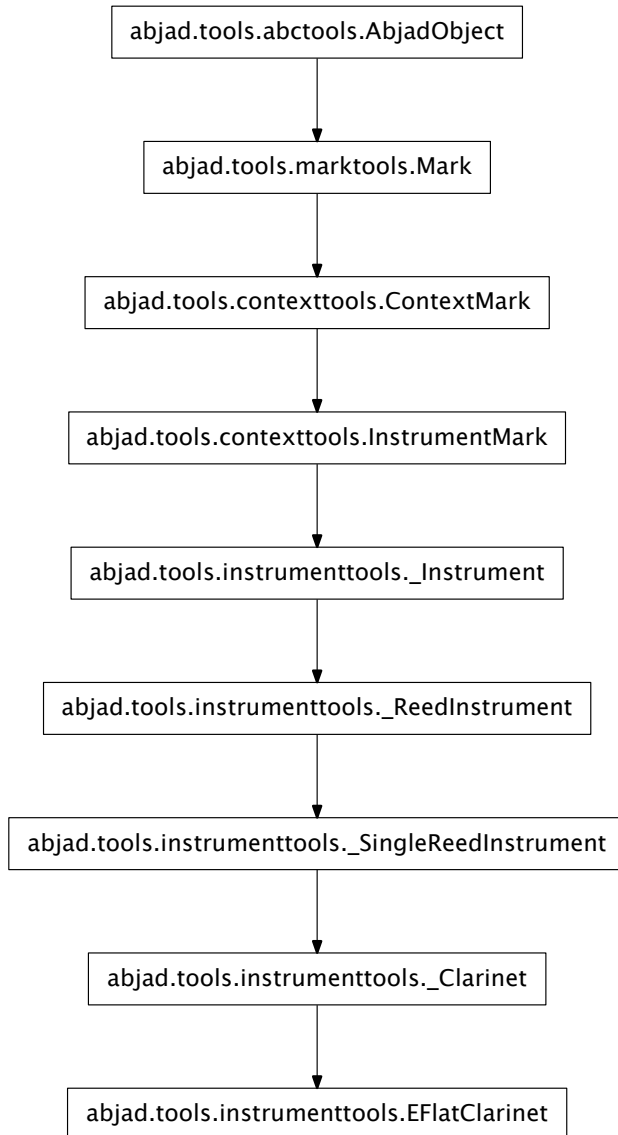
Inherited from `abctools.AbjadObject`

`ContraltoVoice.__ne__(arg)`

Inherited from `marktools.Mark`

`ContraltoVoice.__repr__()`

Inherited from `marktools.Mark`

instrumenttools.EFlatClarinet

class instrumenttools.**EFlatClarinet** (**kwargs)

New in version 2.0. Abjad model of the E-flat clarinet:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> instrumenttools.EFlatClarinet()(staff)
EFlatClarinet()(Staff{4})
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in E-flat }
  \set Staff.shortInstrumentName = \markup { Cl. E-flat }
  c'8
  d'8
  e'8
  f'8
}
```

The E-flat clarinet targets staff context by default.

Read-only Properties

EFlatClarinet.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

EFlatClarinet.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

EFlatClarinet.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

EFlatClarinet.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

EFlatClarinet.is_primary_instrument

Inherited from `instrumenttools._Instrument`

EFlatClarinet.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

EFlatClarinet.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

EFlatClarinet.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`EFlatClarinet.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`EFlatClarinet.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`EFlatClarinet.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

`EFlatClarinet.all_clefs`

Inherited from `instrumenttools._Instrument`

`EFlatClarinet.instrument_name`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.pitch_range`

Inherited from `instrumenttools._Instrument`

`EFlatClarinet.primary_clefs`

Inherited from `instrumenttools._Instrument`

`EFlatClarinet.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`EFlatClarinet.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`EFlatClarinet.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`EFlatClarinet.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`EFlatClarinet.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`EFlatClarinet.__call__(*args)`

Inherited from `marktools.Mark`

`EFlatClarinet.__delattr__(*args)`

Inherited from `marktools.Mark`

`EFlatClarinet.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`EFlatClarinet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`


```
EFlatClarinet.__hash__()
```

Inherited from `contexttools.InstrumentMark`

```
EFlatClarinet.__le__(arg)
```

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

```
EFlatClarinet.__lt__(arg)
```

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

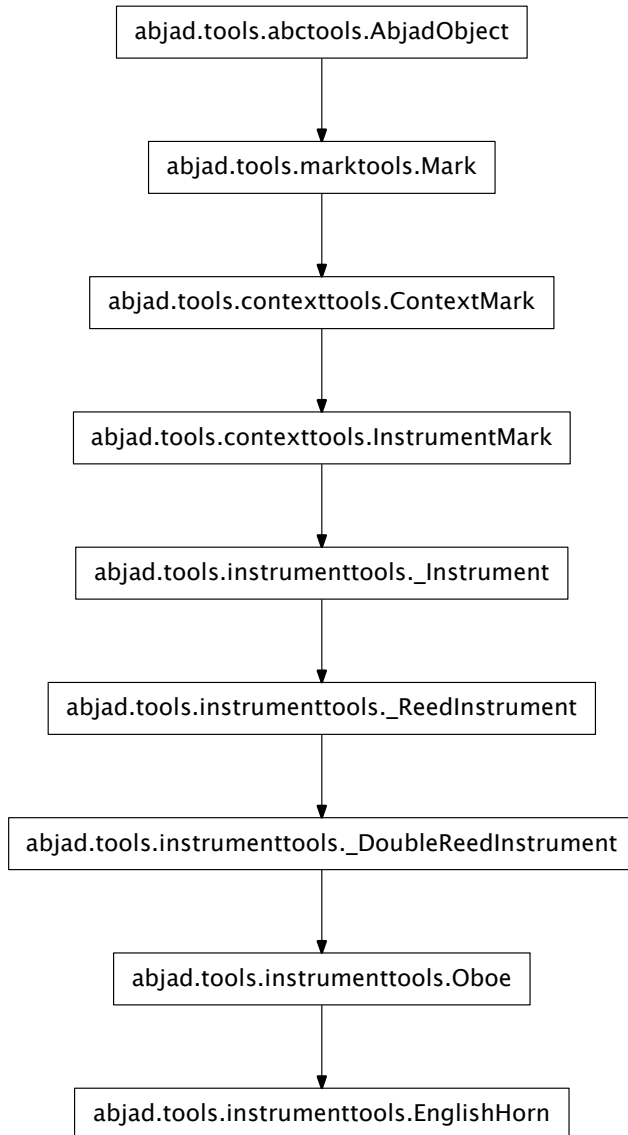
```
EFlatClarinet.__ne__(arg)
```

Inherited from `marktools.Mark`

```
EFlatClarinet.__repr__()
```

Inherited from `marktools.Mark`

instrumenttools.EnglishHorn



```

class instrumenttools.EnglishHorn(**kwargs)
    New in version 2.0. Abjad model of the English horn:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.EnglishHorn()(staff)
    EnglishHorn()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { English horn }
  \set Staff.shortInstrumentName = \markup { Eng. hn. }
  c'8
  d'8
  e'8
  f'8
}
```

The English horn targets staff context by default.

Read-only Properties

EnglishHorn.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

EnglishHorn.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

EnglishHorn.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

EnglishHorn.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

EnglishHorn.is_primary_instrument

Inherited from `instrumenttools._Instrument`

EnglishHorn.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

EnglishHorn.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

EnglishHorn.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

EnglishHorn.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

EnglishHorn.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

EnglishHorn.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

EnglishHorn.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

EnglishHorn.all_clefs

Inherited from `instrumenttools._Instrument`

EnglishHorn.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`EnglishHorn.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`EnglishHorn.pitch_range`

Inherited from `instrumenttools._Instrument`

`EnglishHorn.primary_clefs`

Inherited from `instrumenttools._Instrument`

`EnglishHorn.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`EnglishHorn.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

EnglishHorn.**sounding_pitch_of_fingered_middle_c**

Inherited from `instrumenttools._Instrument`

Methods

EnglishHorn.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

EnglishHorn.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

EnglishHorn.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

EnglishHorn.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

EnglishHorn.**__call__**(*args)

Inherited from `marktools.Mark`

EnglishHorn.**__delattr__**(*args)

Inherited from `marktools.Mark`

EnglishHorn.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

EnglishHorn.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

EnglishHorn.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

EnglishHorn.__hash__()
Inherited from contexttools.InstrumentMark

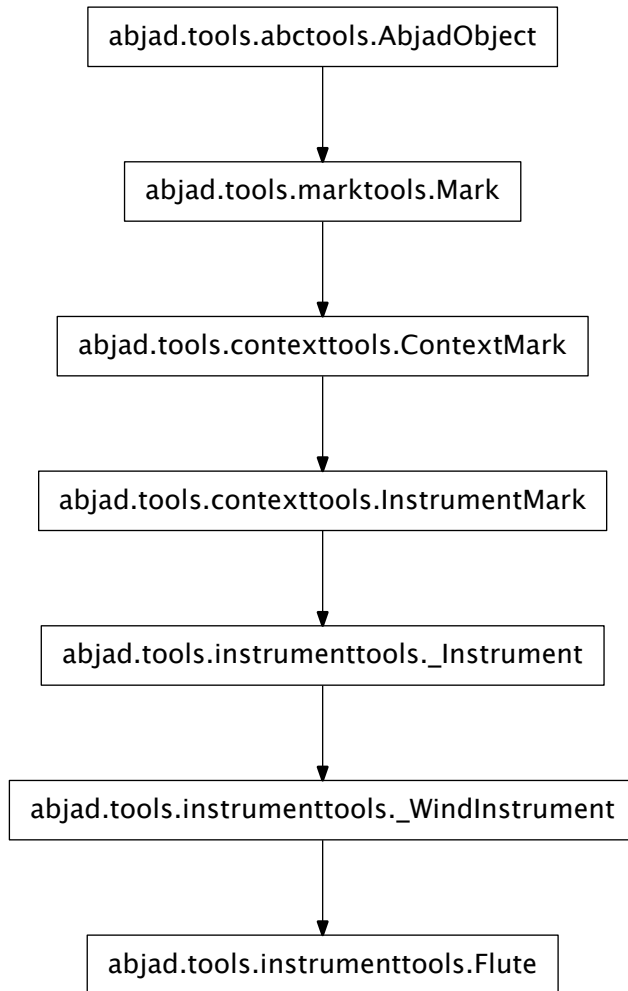
EnglishHorn.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from abctools.AbjadObject

EnglishHorn.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from abctools.AbjadObject

EnglishHorn.__ne__(arg)
Inherited from marktools.Mark

EnglishHorn.__repr__()
Inherited from marktools.Mark

instrumenttools.Flute



class instrumenttools.**Flute**(**kwargs)

New in version 2.0. Abjad model of the flute:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> instrumenttools.Flute()(staff)
Flute()(Staff{4})
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Flute }
  \set Staff.shortInstrumentName = \markup { Fl. }
  c'8
  d'8
```



```
e'8
f'8
}
```

The flute targets staff context by default.

Read-only Properties

Flute.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Flute.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Flute.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Flute.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Flute.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Flute.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Flute.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Flute.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\\set Staff.instrumentName = \\markup { Flute }',
 '\\set Staff.shortInstrumentName = \\markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Flute.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Flute.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Flute.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Flute.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Flute.all_clefs

Inherited from `instrumenttools._Instrument`

Flute.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Flute.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Flute.pitch_range

Inherited from `instrumenttools._Instrument`

Flute.primary_clefs

Inherited from `instrumenttools._Instrument`

Flute.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Flute.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Flute.sounding_pitch_of_fingered_middle_c

Inherited from `instrumenttools._Instrument`

Methods

Flute.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Flute.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Flute.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Flute.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Flute.**__call__**(*args)

Inherited from `marktools.Mark`

Flute.**__delattr__**(*args)

Inherited from `marktools.Mark`

Flute.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

Flute.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Flute.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Flute.**__hash__**()

Inherited from `contexttools.InstrumentMark`

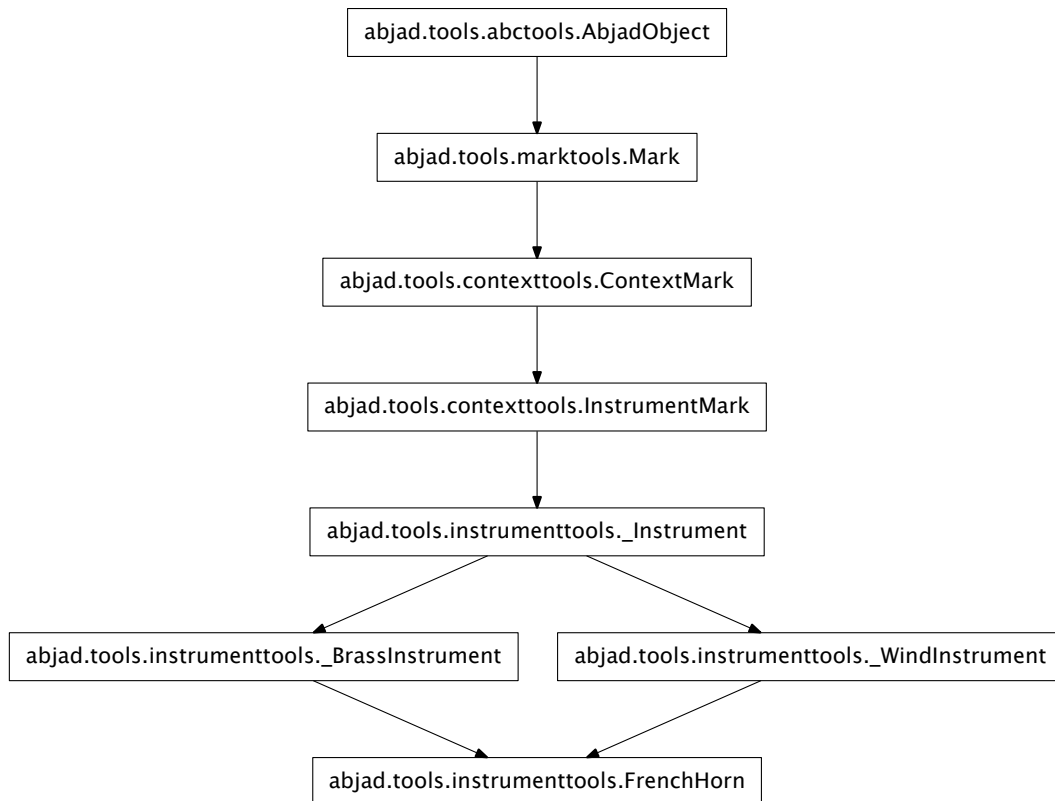
`Flute.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Flute.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Flute.__ne__(arg)`
 Inherited from `marktools.Mark`

`Flute.__repr__()`
 Inherited from `marktools.Mark`

instrumenttools.FrenchHorn



class `instrumenttools.FrenchHorn` (**kwargs)
 New in version 2.0. Abjad model of the French horn:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> instrumenttools.FrenchHorn()(staff)
FrenchHorn()(Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Horn }
  \set Staff.shortInstrumentName = \markup { Hn. }
  c'8
  d'8
  e'8
  f'8
}
```

The French horn targets staff context by default.

Read-only Properties

FrenchHorn.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

FrenchHorn.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

FrenchHorn.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

FrenchHorn.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

FrenchHorn.is_primary_instrument

Inherited from `instrumenttools._Instrument`

FrenchHorn.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

FrenchHorn.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

FrenchHorn.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

FrenchHorn.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

FrenchHorn.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

FrenchHorn.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

FrenchHorn.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

FrenchHorn.all_clefs

Inherited from `instrumenttools._Instrument`

FrenchHorn.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

FrenchHorn.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

FrenchHorn.pitch_range

Inherited from `instrumenttools._Instrument`

FrenchHorn.primary_clefs

Inherited from `instrumenttools._Instrument`

FrenchHorn.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

FrenchHorn.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```


Return markup.

Inherited from `contexttools.InstrumentMark`

`FrenchHorn.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`FrenchHorn.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`FrenchHorn.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`FrenchHorn.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`FrenchHorn.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`FrenchHorn.__call__(*args)`

Inherited from `marktools.Mark`

`FrenchHorn.__delattr__(*args)`

Inherited from `marktools.Mark`

`FrenchHorn.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`FrenchHorn.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

FrenchHorn.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

FrenchHorn.__hash__()

Inherited from `contexttools.InstrumentMark`

FrenchHorn.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

FrenchHorn.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

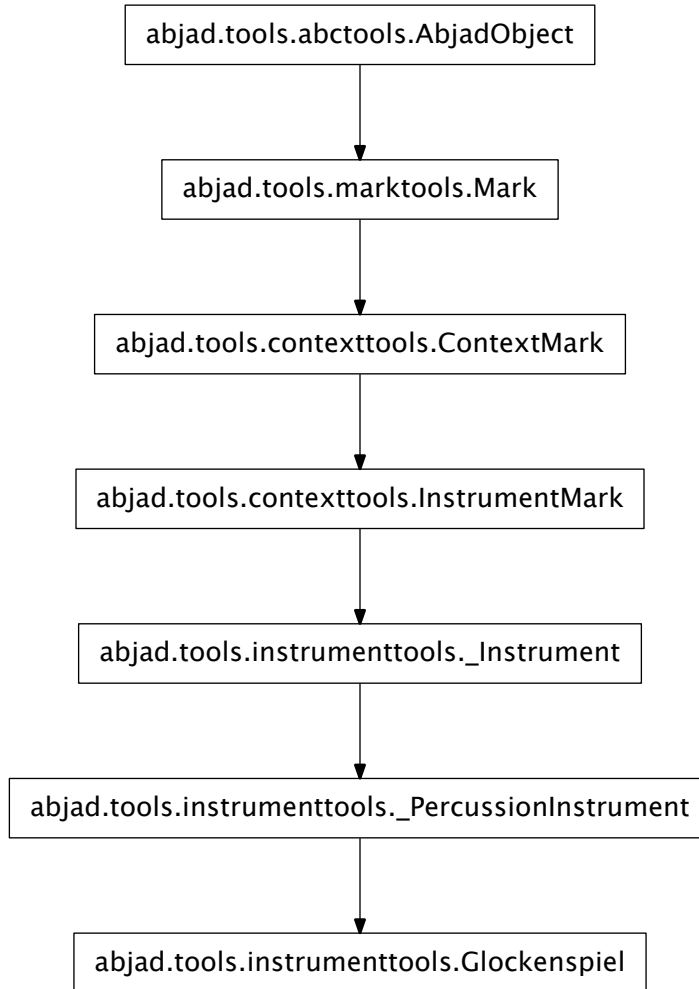
Inherited from `abctools.AbjadObject`

FrenchHorn.__ne__(arg)

Inherited from `marktools.Mark`

FrenchHorn.__repr__()

Inherited from `marktools.Mark`

instrumenttools.Glockenspiel

```

class instrumenttools.Glockenspiel(**kwargs)
    New in version 2.0. Abjad model of the glockenspiel:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.Glockenspiel()(staff)
    Glockenspiel()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Glockenspiel }
      \set Staff.shortInstrumentName = \markup { Gkspl. }
      c'8
      d'8
  
```

```

        e' 8
        f' 8
    }

```

The `glockenspiel` targets staff context by default.

Read-only Properties

`Glockenspiel.default_instrument_name`

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`Glockenspiel.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`Glockenspiel.effective_context`

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`Glockenspiel.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

`Glockenspiel.is_primary_instrument`

Inherited from `instrumenttools._Instrument`

`Glockenspiel.is_secondary_instrument`

Inherited from `instrumenttools._Instrument`

`Glockenspiel.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

`Glockenspiel.lilypond_format`

Read-only LilyPond input format of instrument mark:

```

>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Inherited from `contexttools.InstrumentMark`

`Glockenspiel.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`Glockenspiel.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`Glockenspiel.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`Glockenspiel.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

`Glockenspiel.all_clefs`

Inherited from `instrumenttools._Instrument`

`Glockenspiel.instrument_name`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`Glockenspiel.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`Glockenspiel.pitch_range`

Inherited from `instrumenttools._Instrument`

`Glockenspiel.primary_clefs`

Inherited from `instrumenttools._Instrument`

`Glockenspiel.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`Glockenspiel.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`Glockenspiel.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

Glockenspiel.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Glockenspiel.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Glockenspiel.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Glockenspiel.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Glockenspiel.**__call__**(*args)

Inherited from `marktools.Mark`

Glockenspiel.**__delattr__**(*args)

Inherited from `marktools.Mark`

Glockenspiel.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

Glockenspiel.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Glockenspiel.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Glockenspiel.**__hash__**()

Inherited from `contexttools.InstrumentMark`

`Glockenspiel.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Glockenspiel.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

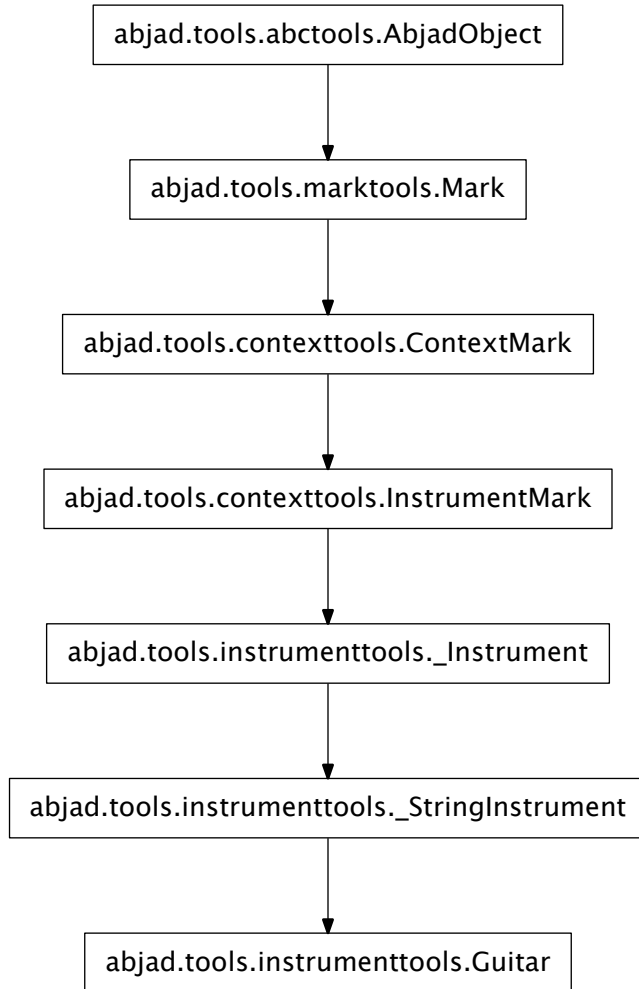
Inherited from `abctools.AbjadObject`

`Glockenspiel.__ne__(arg)`

Inherited from `marktools.Mark`

`Glockenspiel.__repr__()`

Inherited from `marktools.Mark`

instrumenttools.Guitar

```

class instrumenttools.Guitar(**kwargs)
    New in version 2.0. Abjad model of the guitar:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.Guitar()(staff)
    Guitar()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Guitar }
      \set Staff.shortInstrumentName = \markup { Gt. }
      c'8
      d'8
  
```

```

        e' 8
        f' 8
    }

```

The guitar targets staff context by default.

Read-only Properties

Guitar.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Guitar.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Guitar.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Guitar.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Guitar.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Guitar.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Guitar.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Guitar.lilypond_format

Read-only LilyPond input format of instrument mark:

```

>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\\set Staff.instrumentName = \\markup { Flute }',
 '\\set Staff.shortInstrumentName = \\markup { Fl. }']

```

Return list.

Inherited from `contexttools.InstrumentMark`

Guitar.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Guitar.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Guitar.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Guitar.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Guitar.all_clefs

Inherited from `instrumenttools._Instrument`

Guitar.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Guitar.**instrument_name_markup**

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Guitar.**pitch_range**

Inherited from instrumenttools._Instrument

Guitar.**primary_clefs**

Inherited from instrumenttools._Instrument

Guitar.**short_instrument_name**

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Guitar.**short_instrument_name_markup**

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Guitar.**sounding_pitch_of_fingered_middle_c**

Inherited from instrumenttools._Instrument

Methods

Guitar.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Guitar.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Guitar.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Guitar.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Guitar.**__call__**(*args)

Inherited from `marktools.Mark`

Guitar.**__delattr__**(*args)

Inherited from `marktools.Mark`

Guitar.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

Guitar.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Guitar.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Guitar.**__hash__**()

Inherited from `contexttools.InstrumentMark`

Guitar.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Guitar.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

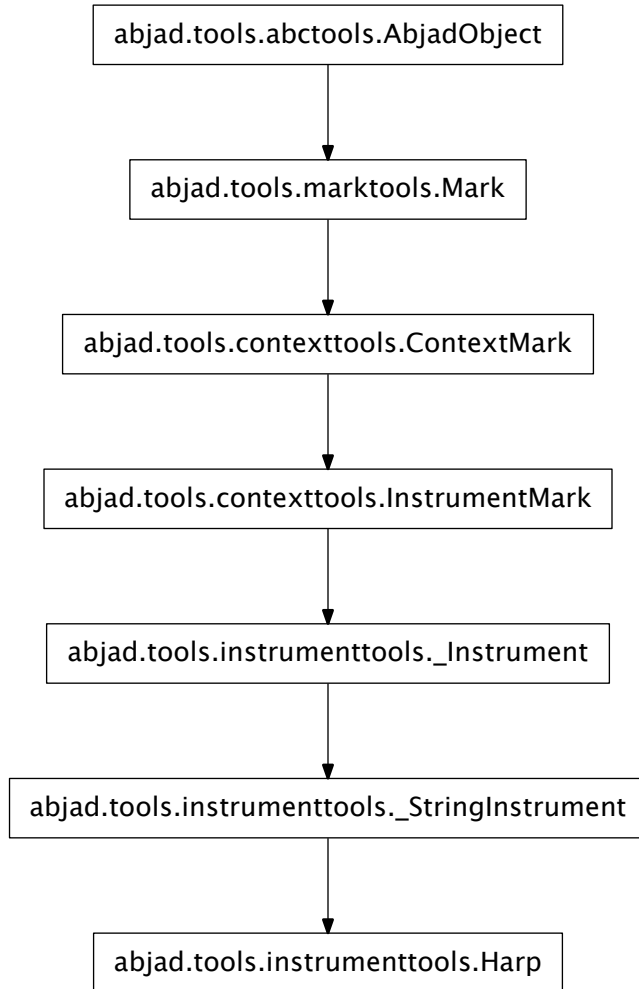
Inherited from `abctools.AbjadObject`

Guitar.__ne__(arg)

Inherited from `marktools.Mark`

Guitar.__repr__()

Inherited from `marktools.Mark`

instrumenttools.Harp

class instrumenttools.**Harp**(target_context=None, **kwargs)

New in version 2.0. Abjad model of the harp:

```
>>> piano_staff = scoretools.PianoStaff([Staff("c'8 d'8 e'8 f'8"), Staff("c'4 b4")])
```

```
>>> instrumenttools.Harp()(piano_staff)
Harp()(PianoStaff<<2>>)
```

```
>>> f(piano_staff)
\\new PianoStaff <<
  \\set PianoStaff.instrumentName = \\markup { Harp }
  \\set PianoStaff.shortInstrumentName = \\markup { Hp. }
  \\new Staff {
    c'8
```

```

        d'8
        e'8
        f'8
    }
    \new Staff {
        c'4
        b4
    }
>>

```

The harp targets piano staff context by default.

Read-only Properties

Harp.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Harp.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Harp.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Harp.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Harp.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Harp.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Harp.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Harp.lilypond_format

Read-only LilyPond input format of instrument mark:


```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Harp.**start_component**

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Harp.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Harp.**target_context**

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Harp.**traditional_pitch_range**

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Harp.**all_clefs**

Inherited from `instrumenttools._Instrument`

Harp.**instrument_name**

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Harp.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Harp.pitch_range

Inherited from `instrumenttools._Instrument`

Harp.primary_clefs

Inherited from `instrumenttools._Instrument`

Harp.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Harp.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Harp.**sounding_pitch_of_fingered_middle_c**
 Inherited from `instrumenttools._Instrument`

Methods

Harp.**attach**(*start_component*)
 Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Harp.**detach**()
 Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Harp.**get_default_performer_name**(*locale=None*)
 New in version 2.5. Get default player name.
 Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Harp.**get_performer_names**()
 New in version 2.5. Get performer names.
 Inherited from `instrumenttools._Instrument`

Special Methods

Harp.**__call__**(*args)
 Inherited from `marktools.Mark`

Harp.**__delattr__**(*args)
 Inherited from `marktools.Mark`

Harp.**__eq__**(arg)
 Inherited from `contexttools.InstrumentMark`

Harp.**__ge__**(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

Harp.**__gt__**(arg)
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

Harp. **__hash__**()
Inherited from `contexttools.InstrumentMark`

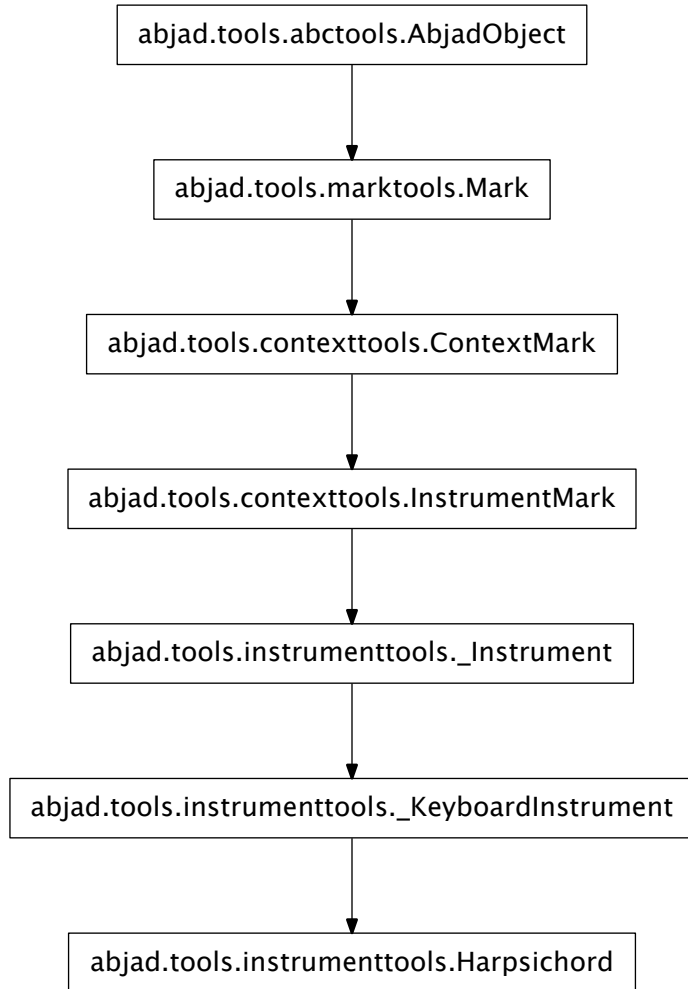
Harp. **__le__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

Harp. **__lt__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

Harp. **__ne__**(*arg*)
Inherited from `marktools.Mark`

Harp. **__repr__**()
Inherited from `marktools.Mark`

instrumenttools.Harpsichord



class instrumenttools.**Harpsichord**(target_context=None, **kwargs)

New in version 2.5. Abjad model of the harpsichord:

```
>>> piano_staff = scoretools.PianoStaff([Staff("c'8 d'8 e'8 f'8"), Staff("c'4 b4")])
```

```
>>> instrumenttools.Harpsichord()(piano_staff)
Harpsichord() (PianoStaff<<2>>)
```

```
>>> f(piano_staff)
\new PianoStaff <<
  \set PianoStaff.instrumentName = \markup { Harpsichord }
  \set PianoStaff.shortInstrumentName = \markup { Hpschd. }
  \new Staff {
    c'8
```

```

        d'8
        e'8
        f'8
    }
    \new Staff {
        c'4
        b4
    }
>>

```

The harpsichord targets piano staff context by default.

Return instrument.

Read-only Properties

Harpsichord.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Harpsichord.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Harpsichord.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Harpsichord.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Harpsichord.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Harpsichord.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Harpsichord.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Harpsichord.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Harpsichord.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Harpsichord.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Harpsichord.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Harpsichord.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Harpsichord.all_clefs

Inherited from `instrumenttools._Instrument`

Harpsichord.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Harpsichord.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Harpsichord.pitch_range

Inherited from `instrumenttools._Instrument`

Harpsichord.primary_clefs

Inherited from `instrumenttools._Instrument`

Harpsichord.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Harpsichord.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Harpsichord.**sounding_pitch_of_fingered_middle_c**

Inherited from instrumenttools._Instrument

Methods

Harpsichord.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from contexttools.ContextMark

Harpsichord.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from contexttools.ContextMark

Harpsichord.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from instrumenttools._Instrument

Harpsichord.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from instrumenttools._Instrument

Special Methods

Harpsichord.**__call__**(*args)

Inherited from marktools.Mark

Harpsichord.**__delattr__**(*args)

Inherited from marktools.Mark

Harpsichord.**__eq__**(arg)

Inherited from contexttools.InstrumentMark

Harpsichord.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

Harpsichord.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

`Harpsichord.__hash__()`
 Inherited from `contexttools.InstrumentMark`

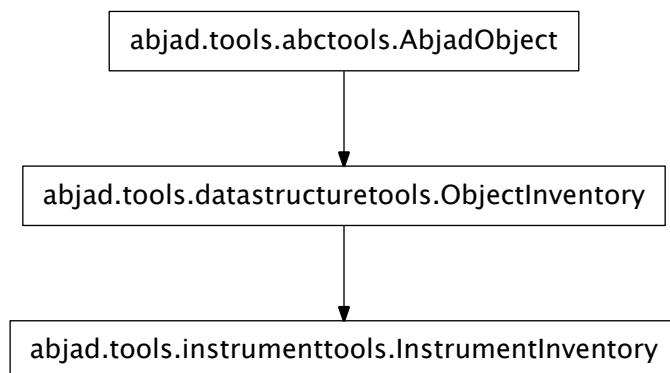
`Harpsichord.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Harpsichord.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Harpsichord.__ne__(arg)`
 Inherited from `marktools.Mark`

`Harpsichord.__repr__()`
 Inherited from `marktools.Mark`

instrumenttools.InstrumentInventory



class `instrumenttools.InstrumentInventory` (*tokens=None, name=None*)
 New in version 2.8. Abjad model of an ordered list of instruments:

```

>>> inventory = instrumenttools.InstrumentInventory(
...     [instrumenttools.Flute(), instrumenttools.Guitar()])

>>> inventory
InstrumentInventory([Flute(), Guitar()])
    
```

Instrument inventories implement list interface and are mutable.

Read-only Properties

`InstrumentInventory.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`InstrumentInventory.name`

Read / write name of inventory.

Inherited from `datastructuretools.ObjectInventory`

Methods

`InstrumentInventory.append(token)`

Change *token* to item and append.

Inherited from `datastructuretools.ObjectInventory`

`InstrumentInventory.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`InstrumentInventory.extend(tokens)`

Change *tokens* to items and extend.

Inherited from `datastructuretools.ObjectInventory`

`InstrumentInventory.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`InstrumentInventory.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`InstrumentInventory.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`InstrumentInventory.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`InstrumentInventory.reverse()`

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`InstrumentInventory.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`InstrumentInventory.__add__()`

`x.__add__(y)` <==> `x+y`

Inherited from `__builtin__.list`

`InstrumentInventory.__contains__(token)`
 Inherited from `datastructuretools.ObjectInventory`

`InstrumentInventory.__delitem__()`
`x.__delitem__(y) <==> del x[y]`
 Inherited from `__builtin__.list`

`InstrumentInventory.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

`InstrumentInventory.__eq__()`
`x.__eq__(y) <==> x==y`
 Inherited from `__builtin__.list`

`InstrumentInventory.__ge__()`
`x.__ge__(y) <==> x>=y`
 Inherited from `__builtin__.list`

`InstrumentInventory.__getitem__()`
`x.__getitem__(y) <==> x[y]`
 Inherited from `__builtin__.list`

`InstrumentInventory.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

`InstrumentInventory.__gt__()`
`x.__gt__(y) <==> x>y`
 Inherited from `__builtin__.list`

`InstrumentInventory.__iadd__()`
`x.__iadd__(y) <==> x+=y`
 Inherited from `__builtin__.list`

`InstrumentInventory.__imul__()`
`x.__imul__(y) <==> x*=y`
 Inherited from `__builtin__.list`

`InstrumentInventory.__iter__() <==> iter(x)`
 Inherited from `__builtin__.list`

`InstrumentInventory.__le__()`
`x.__le__(y) <==> x<=y`
 Inherited from `__builtin__.list`

`InstrumentInventory.__len__() <==> len(x)`
 Inherited from `__builtin__.list`

`InstrumentInventory.__lt__()`
`x.__lt__(y) <==> x<y`
 Inherited from `__builtin__.list`

```
InstrumentInventory.__mul__()
    x.__mul__(n) <==> x*n

    Inherited from __builtin__.list

InstrumentInventory.__ne__()
    x.__ne__(y) <==> x!=y

    Inherited from __builtin__.list

InstrumentInventory.__repr__()
    Inherited from datastructuretools.ObjectInventory

InstrumentInventory.__reversed__()
    L.__reversed__() – return a reverse iterator over the list

    Inherited from __builtin__.list

InstrumentInventory.__rmul__()
    x.__rmul__(n) <==> n*x

    Inherited from __builtin__.list

InstrumentInventory.__setitem__()
    x.__setitem__(i, y) <==> x[i]=y

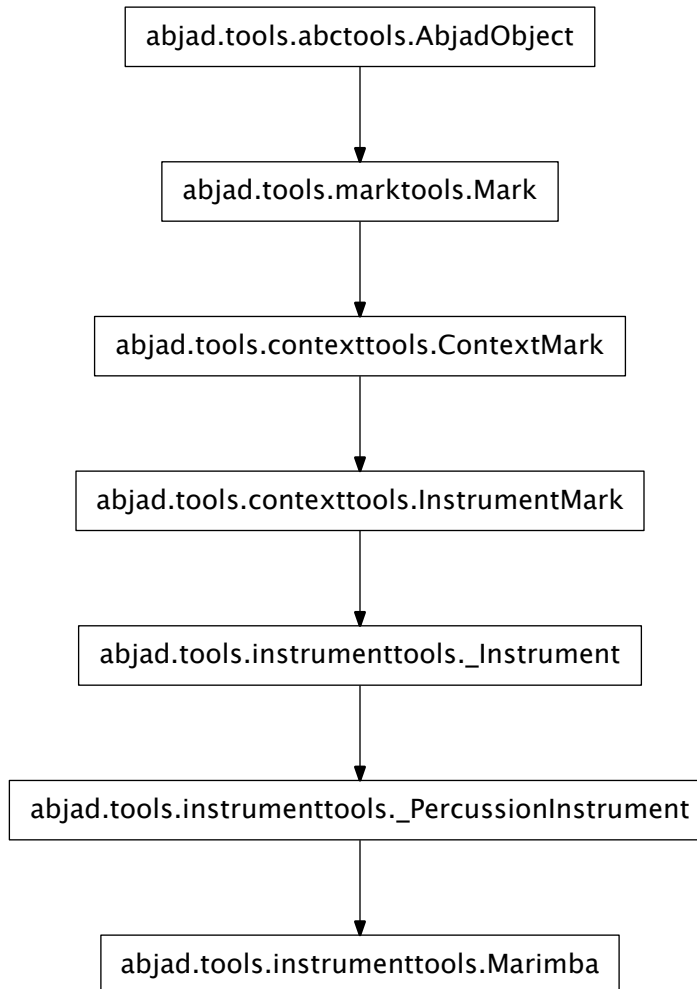
    Inherited from __builtin__.list

InstrumentInventory.__setslice__()
    x.__setslice__(i, j, y) <==> x[i:j]=y

    Use of negative indices is not supported.

    Inherited from __builtin__.list
```

instrumenttools.Marimba



class instrumenttools.**Marimba** *(**kwargs)*
 New in version 2.0. Abjad model of the marimba:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> instrumenttools.Marimba() (staff)
Marimba() (Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Marimba }
  \set Staff.shortInstrumentName = \markup { Mb. }
  c'8
  d'8

```

```
e' 8
f' 8
}
```

The marimba targets staff context by default.

Read-only Properties

Marimba.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Marimba.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Marimba.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Marimba.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Marimba.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Marimba.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Marimba.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Marimba.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\\set Staff.instrumentName = \\markup { Flute }',
 '\\set Staff.shortInstrumentName = \\markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Marimba.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Marimba.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Marimba.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Marimba.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Marimba.all_clefs

Inherited from `instrumenttools._Instrument`

Marimba.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Marimba.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Marimba.pitch_range

Inherited from instrumenttools._Instrument

Marimba.primary_clefs

Inherited from instrumenttools._Instrument

Marimba.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Marimba.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Marimba.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

Marimba.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Marimba.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Marimba.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Marimba.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Marimba.**__call__**(*args)

Inherited from `marktools.Mark`

Marimba.**__delattr__**(*args)

Inherited from `marktools.Mark`

Marimba.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

Marimba.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Marimba.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Marimba.**__hash__**()

Inherited from `contexttools.InstrumentMark`

Marimba.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Marimba.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

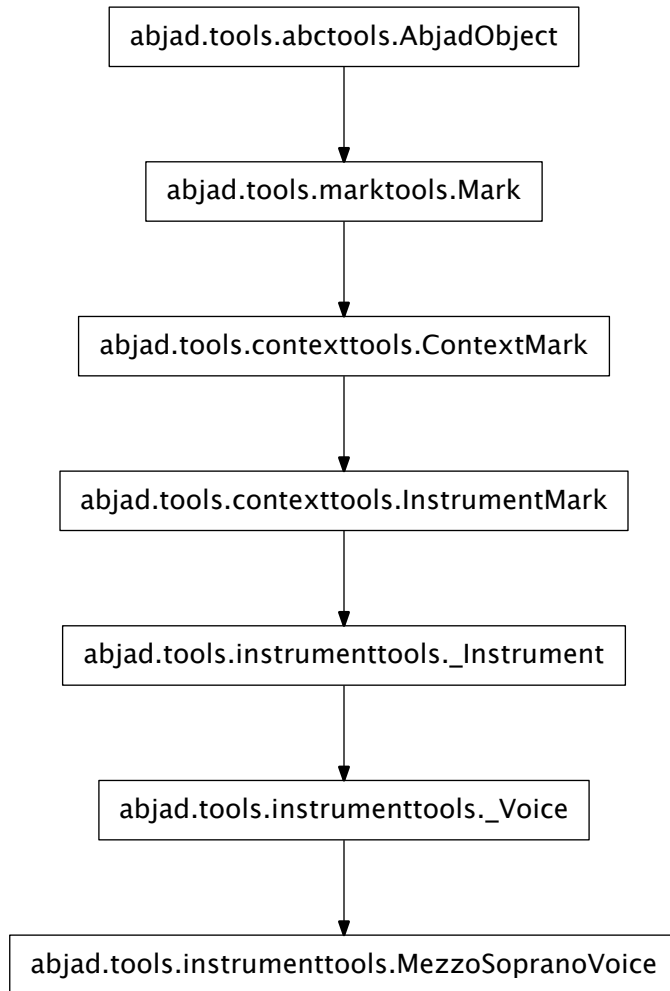
Marimba.__ne__(arg)

Inherited from `marktools.Mark`

Marimba.__repr__()

Inherited from `marktools.Mark`

instrumenttools.MezzoSopranoVoice



```

class instrumenttools.MezzoSopranoVoice(**kwargs)
    New in version 2.8. Abjad model of the mezzo-soprano voice:

    >>> staff = Staff("c''8 d''8 e''8 f''8")

    >>> instrumenttools.MezzoSopranoVoice()(staff)
    MezzoSopranoVoice()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Mezzo-soprano voice }
      \set Staff.shortInstrumentName = \markup { Mezzo-soprano }
      c''8
      d''8
  
```

```
e''8
f''8
}
```

The mezzo-soprano voice targets staff context by default.

Read-only Properties

MezzoSopranoVoice.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

MezzoSopranoVoice.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

MezzoSopranoVoice.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

MezzoSopranoVoice.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

MezzoSopranoVoice.is_primary_instrument

Inherited from `instrumenttools._Instrument`

MezzoSopranoVoice.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

MezzoSopranoVoice.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

MezzoSopranoVoice.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

MezzoSopranoVoice.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

MezzoSopranoVoice.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

MezzoSopranoVoice.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

MezzoSopranoVoice.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

MezzoSopranoVoice.all_clefs

Inherited from `instrumenttools._Instrument`

MezzoSopranoVoice.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.pitch_range`

Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.primary_clefs`

Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`MezzoSopranoVoice.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`MezzoSopranoVoice.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`MezzoSopranoVoice.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`MezzoSopranoVoice.__call__(*args)`

Inherited from `marktools.Mark`

`MezzoSopranoVoice.__delattr__(*args)`

Inherited from `marktools.Mark`

`MezzoSopranoVoice.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MezzoSopranoVoice.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MezzoSopranoVoice.__hash__()`

Inherited from `contexttools.InstrumentMark`

MezzoSopranoVoice.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

MezzoSopranoVoice.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

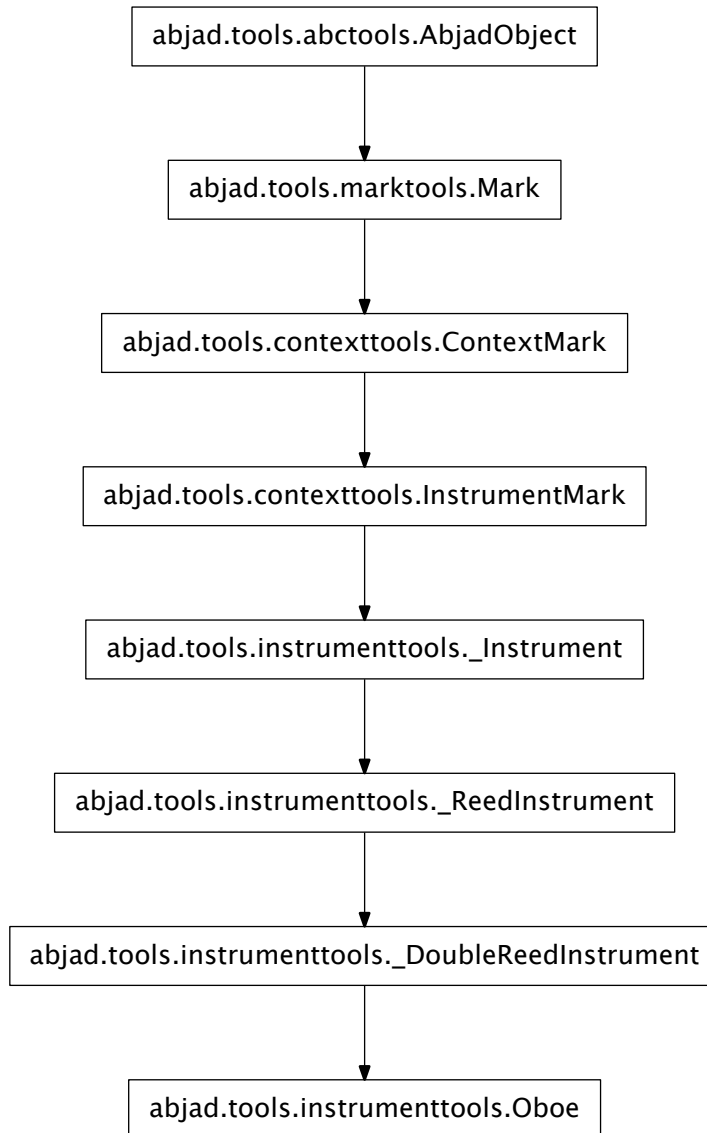
Inherited from `abctools.AbjadObject`

MezzoSopranoVoice.__ne__(arg)

Inherited from `marktools.Mark`

MezzoSopranoVoice.__repr__()

Inherited from `marktools.Mark`

instrumenttools.Oboe

```
class instrumenttools.Oboe (**kwargs)
    New in version 2.0. Abjad model of the oboe:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.Oboe() (staff)
    Oboe() (Staff{4})
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Oboe }
  \set Staff.shortInstrumentName = \markup { Ob. }
  c'8
  d'8
  e'8
  f'8
}
```

The oboe targets staff context by default.

Read-only Properties

Oboe.**default_instrument_name**

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Oboe.**default_short_instrument_name**

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Oboe.**effective_context**

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Oboe.**interval_of_transposition**

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Oboe.**is_primary_instrument**

Inherited from `instrumenttools._Instrument`

Oboe.**is_secondary_instrument**

Inherited from `instrumenttools._Instrument`

Oboe.**is_transposing**

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Oboe.**lilypond_format**

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Oboe.**start_component**

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Oboe.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Oboe.**target_context**

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Oboe.**traditional_pitch_range**

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Oboe.**all_clefs**

Inherited from `instrumenttools._Instrument`

Oboe.**instrument_name**

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Oboe.**instrument_name_markup**

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Oboe.**pitch_range**

Inherited from `instrumenttools._Instrument`

Oboe.**primary_clefs**

Inherited from `instrumenttools._Instrument`

Oboe.**short_instrument_name**

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Oboe.**short_instrument_name_markup**

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Oboe.**sounding_pitch_of_fingered_middle_c**
 Inherited from `instrumenttools._Instrument`

Methods

Oboe.**attach**(*start_component*)
 Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Oboe.**detach**()
 Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Oboe.**get_default_performer_name**(*locale=None*)
 New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Oboe.**get_performer_names**()
 New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Oboe.**__call__**(*args)
 Inherited from `marktools.Mark`

Oboe.**__delattr__**(*args)
 Inherited from `marktools.Mark`

Oboe.**__eq__**(arg)
 Inherited from `contexttools.InstrumentMark`

Oboe.**__ge__**(arg)
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Oboe.**__gt__**(arg)
 Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Oboe.**__hash__**()
 Inherited from contexttools.InstrumentMark

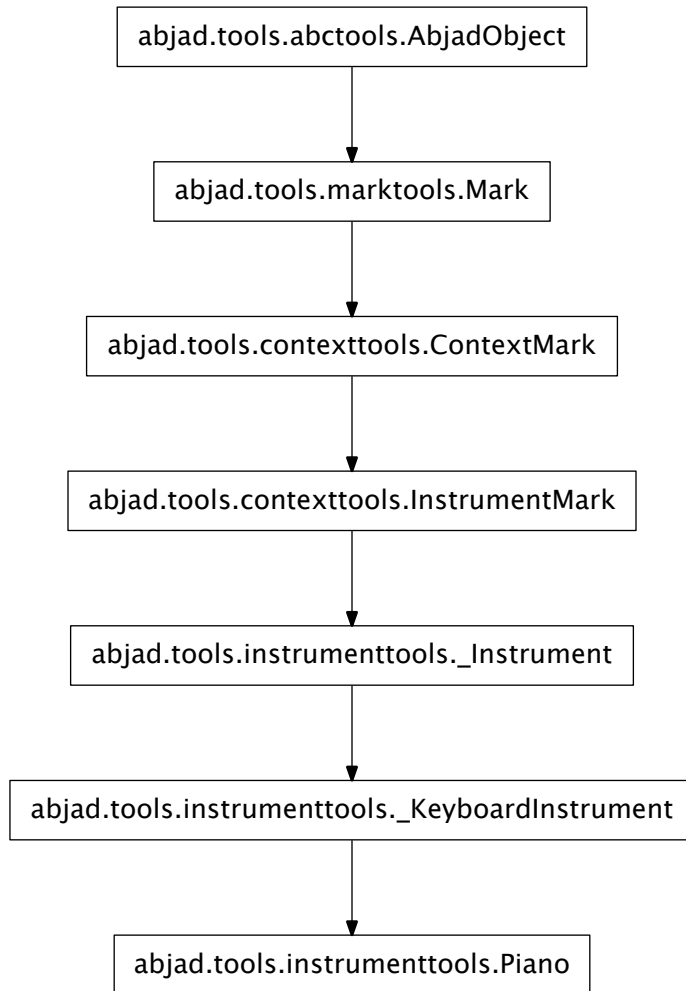
Oboe.**__le__**(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from abctools.AbjadObject

Oboe.**__lt__**(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from abctools.AbjadObject

Oboe.**__ne__**(arg)
 Inherited from marktools.Mark

Oboe.**__repr__**()
 Inherited from marktools.Mark

instrumenttools.Piano



class instrumenttools.**Piano**(target_context=None, **kwargs)

New in version 2.0. Abjad model of the piano:

```
>>> piano_staff = scoretools.PianoStaff([Staff("c'8 d'8 e'8 f'8"), Staff("c'4 b4")])
```

```
>>> instrumenttools.Piano()(piano_staff)
Piano()(PianoStaff<<2>>)
```

```
>>> f(piano_staff)
\\new PianoStaff <<
  \\set PianoStaff.instrumentName = \\markup { Piano }
  \\set PianoStaff.shortInstrumentName = \\markup { Pf. }
  \\new Staff {
    c'8
```



```

        d'8
        e'8
        f'8
    }
    \new Staff {
        c'4
        b4
    }
>>

```

The piano targets piano staff context by default.

Read-only Properties

Piano.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Piano.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Piano.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Piano.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Piano.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Piano.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Piano.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Piano.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Piano.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Piano.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Piano.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Piano.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Piano.all_clefs

Inherited from `instrumenttools._Instrument`

Piano.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Piano.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Piano.pitch_range

Inherited from `instrumenttools._Instrument`

Piano.primary_clefs

Inherited from `instrumenttools._Instrument`

Piano.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Piano.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Piano.**sounding_pitch_of_fingered_middle_c**
 Inherited from `instrumenttools._Instrument`

Methods

Piano.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Piano.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Piano.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Piano.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Piano.**__call__**(*args)

Inherited from `marktools.Mark`

Piano.**__delattr__**(*args)

Inherited from `marktools.Mark`

Piano.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

Piano.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Piano.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Piano.**__hash__**()
Inherited from `contextttools.InstrumentMark`

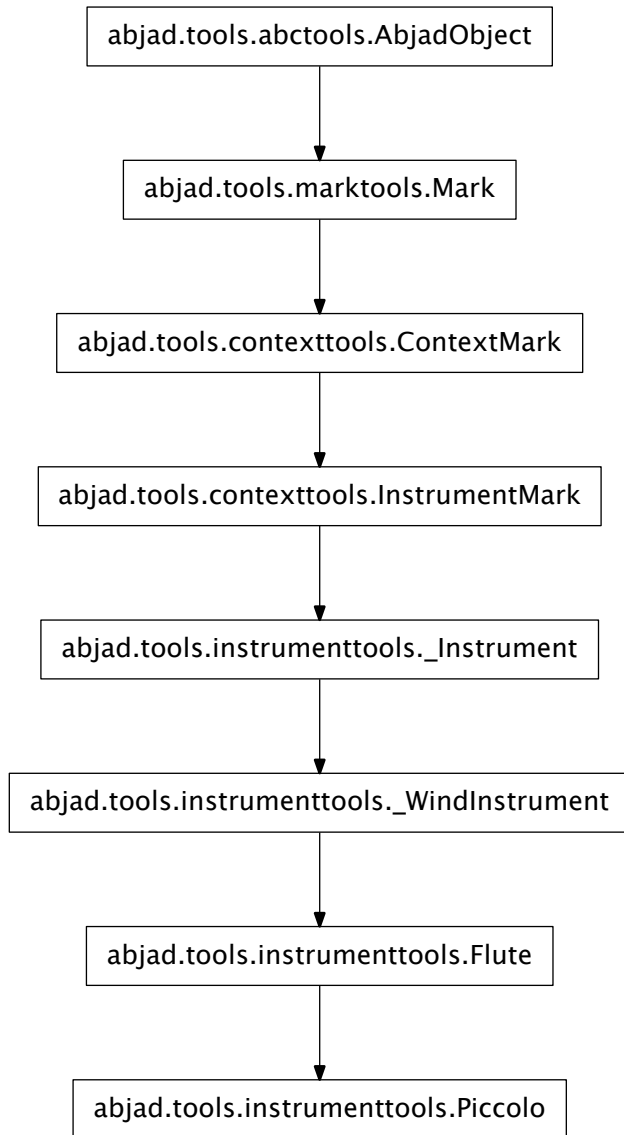
Piano.**__le__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

Piano.**__lt__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

Piano.**__ne__**(*arg*)
Inherited from `marktools.Mark`

Piano.**__repr__**()
Inherited from `marktools.Mark`

instrumenttools.Piccolo



```

class instrumenttools.Piccolo(**kwargs)
    New in version 2.0. Abjad model of the piccolo:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.Piccolo()(staff)
    Piccolo()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Piccolo }
  \set Staff.shortInstrumentName = \markup { Picc. }
  c'8
  d'8
  e'8
  f'8
}
```

The piccolo targets staff context by default.

Read-only Properties

Piccolo.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Piccolo.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Piccolo.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Piccolo.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Piccolo.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Piccolo.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Piccolo.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Piccolo.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Piccolo.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Piccolo.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Piccolo.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Piccolo.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Piccolo.all_clefs

Inherited from `instrumenttools._Instrument`

Piccolo.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:


```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Piccolo.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

Piccolo.pitch_range

Inherited from `instrumenttools._Instrument`

Piccolo.primary_clefs

Inherited from `instrumenttools._Instrument`

Piccolo.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Piccolo.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`Piccolo.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`Piccolo.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`Piccolo.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`Piccolo.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`Piccolo.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`Piccolo.__call__(*args)`

Inherited from `marktools.Mark`

`Piccolo.__delattr__(*args)`

Inherited from `marktools.Mark`

`Piccolo.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`Piccolo.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Piccolo.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Piccolo. **__hash__**()
Inherited from `contextttools.InstrumentMark`

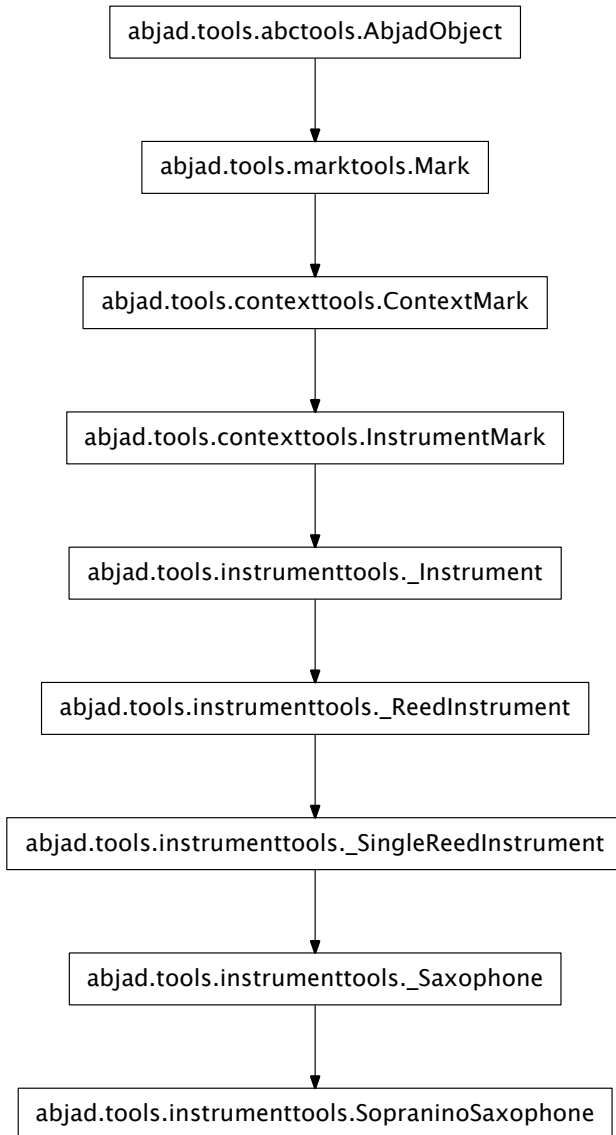
Piccolo. **__le__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

Piccolo. **__lt__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

Piccolo. **__ne__**(*arg*)
Inherited from `marktools.Mark`

Piccolo. **__repr__**()
Inherited from `marktools.Mark`

instrumenttools.SopraninoSaxophone



class `instrumenttools.SopraninoSaxophone` *(**kwargs)*

New in version 2.6. Abjad model of the sopranino saxophone:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> instrumenttools.SopraninoSaxophone()(staff)
SopraninoSaxophone()(Staff{4})
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Sopranino saxophone }
  \set Staff.shortInstrumentName = \markup { Sopranino sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The soprano saxophone is pitched in E-flat.

The soprano saxophone targets staff context by default.

Read-only Properties

SopraninoSaxophone.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

SopraninoSaxophone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

SopraninoSaxophone.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

SopraninoSaxophone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

SopraninoSaxophone.is_primary_instrument

Inherited from `instrumenttools._Instrument`

SopraninoSaxophone.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

SopraninoSaxophone.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

SopraninoSaxophone.**lilypond_format**

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

SopraninoSaxophone.**start_component**

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

SopraninoSaxophone.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

SopraninoSaxophone.**target_context**

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

SopraninoSaxophone.**traditional_pitch_range**

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

SopraninoSaxophone.**all_clefs**

Inherited from `instrumenttools._Instrument`

SopraninoSaxophone.**instrument_name**

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`SopraninoSaxophone.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`SopraninoSaxophone.pitch_range`

Inherited from `instrumenttools._Instrument`

`SopraninoSaxophone.primary_clefs`

Inherited from `instrumenttools._Instrument`

`SopraninoSaxophone.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`SopraninoSaxophone.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`SopraninoSaxophone.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`SopraninoSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`SopraninoSaxophone.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`SopraninoSaxophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`SopraninoSaxophone.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`SopraninoSaxophone.__call__(*args)`

Inherited from `marktools.Mark`

`SopraninoSaxophone.__delattr__(*args)`

Inherited from `marktools.Mark`

`SopraninoSaxophone.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`SopraninoSaxophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SopraninoSaxophone.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`SopraninoSaxophone.__hash__()`
Inherited from `contexttools.InstrumentMark`

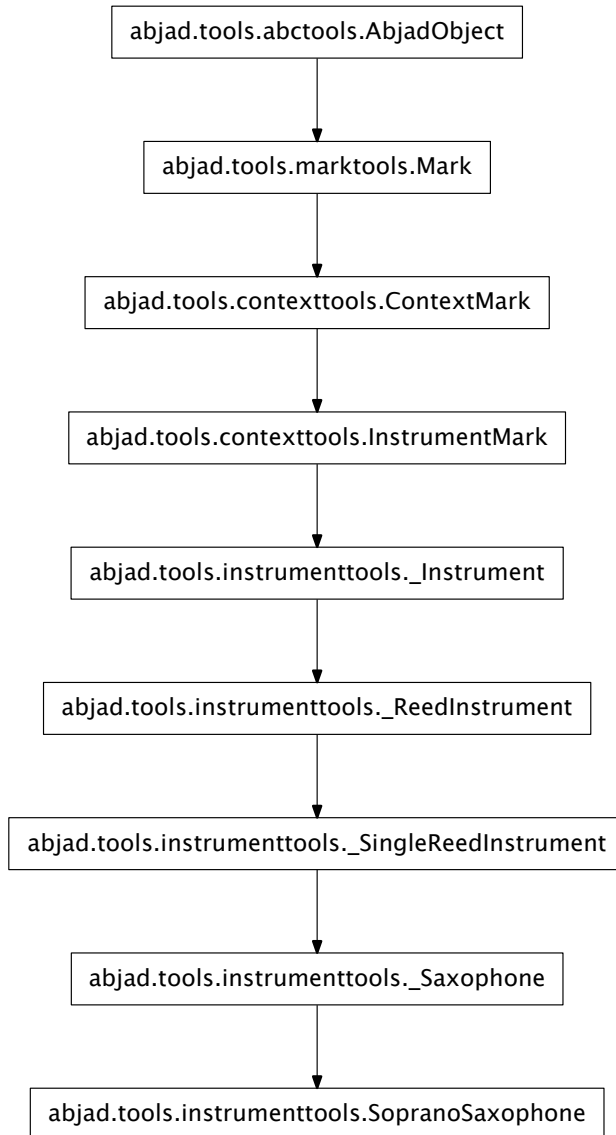
`SopraninoSaxophone.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`SopraninoSaxophone.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`SopraninoSaxophone.__ne__(arg)`
Inherited from `marktools.Mark`

`SopraninoSaxophone.__repr__()`
Inherited from `marktools.Mark`

instrumenttools.SopranoSaxophone



```

class instrumenttools.SopranoSaxophone(**kwargs)
    New in version 2.6. Abjad model of the soprano saxophone:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.SopranoSaxophone()(staff)
    SopranoSaxophone()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Soprano saxophone }
  \set Staff.shortInstrumentName = \markup { Sop. sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The soprano saxophone is pitched in B-flat.

The soprano saxophone targets staff context by default.

Read-only Properties

SopranoSaxophone.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

SopranoSaxophone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

SopranoSaxophone.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

SopranoSaxophone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

SopranoSaxophone.is_primary_instrument

Inherited from `instrumenttools._Instrument`

SopranoSaxophone.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

SopranoSaxophone.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

SopranoSaxophone.**lilypond_format**

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

SopranoSaxophone.**start_component**

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

SopranoSaxophone.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

SopranoSaxophone.**target_context**

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

SopranoSaxophone.**traditional_pitch_range**

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

SopranoSaxophone.**all_clefs**

Inherited from `instrumenttools._Instrument`

SopranoSaxophone.**instrument_name**

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`SopranoSaxophone.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`SopranoSaxophone.pitch_range`

Inherited from `instrumenttools._Instrument`

`SopranoSaxophone.primary_clefs`

Inherited from `instrumenttools._Instrument`

`SopranoSaxophone.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`SopranoSaxophone.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`SopranoSaxophone.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`SopranoSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`SopranoSaxophone.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`SopranoSaxophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`SopranoSaxophone.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`SopranoSaxophone.__call__(*args)`

Inherited from `marktools.Mark`

`SopranoSaxophone.__delattr__(*args)`

Inherited from `marktools.Mark`

`SopranoSaxophone.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`SopranoSaxophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SopranoSaxophone.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`SopranoSaxophone.__hash__()`
Inherited from `contexttools.InstrumentMark`

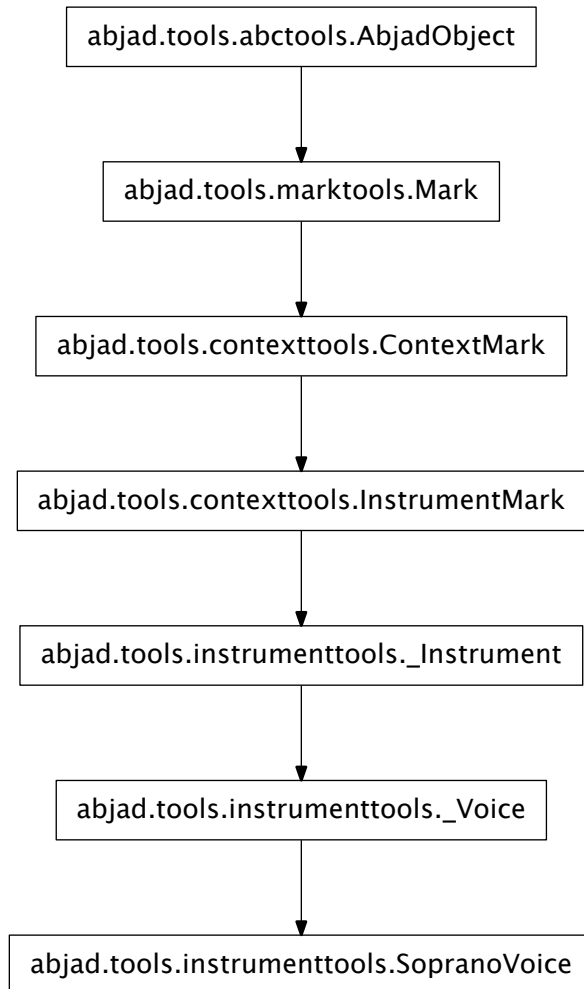
`SopranoSaxophone.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`SopranoSaxophone.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`SopranoSaxophone.__ne__(arg)`
Inherited from `marktools.Mark`

`SopranoSaxophone.__repr__()`
Inherited from `marktools.Mark`

instrumenttools.SopranoVoice



```

class instrumenttools.SopranoVoice(**kwargs)
    New in version 2.8. Abjad model of the soprano voice:

    >>> staff = Staff("c''8 d''8 e''8 f''8")

    >>> instrumenttools.SopranoVoice()(staff)
    SopranoVoice()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Soprano voice }
      \set Staff.shortInstrumentName = \markup { Soprano }
      c''8
      d''8
  
```



```

        e''8
        f''8
    }

```

The soprano voice targets staff context by default.

Read-only Properties

SopranoVoice.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

SopranoVoice.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

SopranoVoice.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

SopranoVoice.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

SopranoVoice.is_primary_instrument

Inherited from `instrumenttools._Instrument`

SopranoVoice.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

SopranoVoice.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

SopranoVoice.lilypond_format

Read-only LilyPond input format of instrument mark:

```

>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Inherited from `contexttools.InstrumentMark`

SopranoVoice.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

SopranoVoice.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

SopranoVoice.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

SopranoVoice.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

SopranoVoice.all_clefs

Inherited from `instrumenttools._Instrument`

SopranoVoice.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

SopranoVoice.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

SopranoVoice.pitch_range

Inherited from instrumenttools._Instrument

SopranoVoice.primary_clefs

Inherited from instrumenttools._Instrument

SopranoVoice.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

SopranoVoice.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

SopranoVoice.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

SopranoVoice.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from contexttools.ContextMark

SopranoVoice.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from contexttools.ContextMark

SopranoVoice.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from instrumenttools._Instrument

SopranoVoice.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from instrumenttools._Instrument

Special Methods

SopranoVoice.**__call__**(*args)

Inherited from marktools.Mark

SopranoVoice.**__delattr__**(*args)

Inherited from marktools.Mark

SopranoVoice.**__eq__**(arg)

Inherited from contexttools.InstrumentMark

SopranoVoice.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

SopranoVoice.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

SopranoVoice.**__hash__**()

Inherited from contexttools.InstrumentMark

SopranoVoice.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

SopranoVoice.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

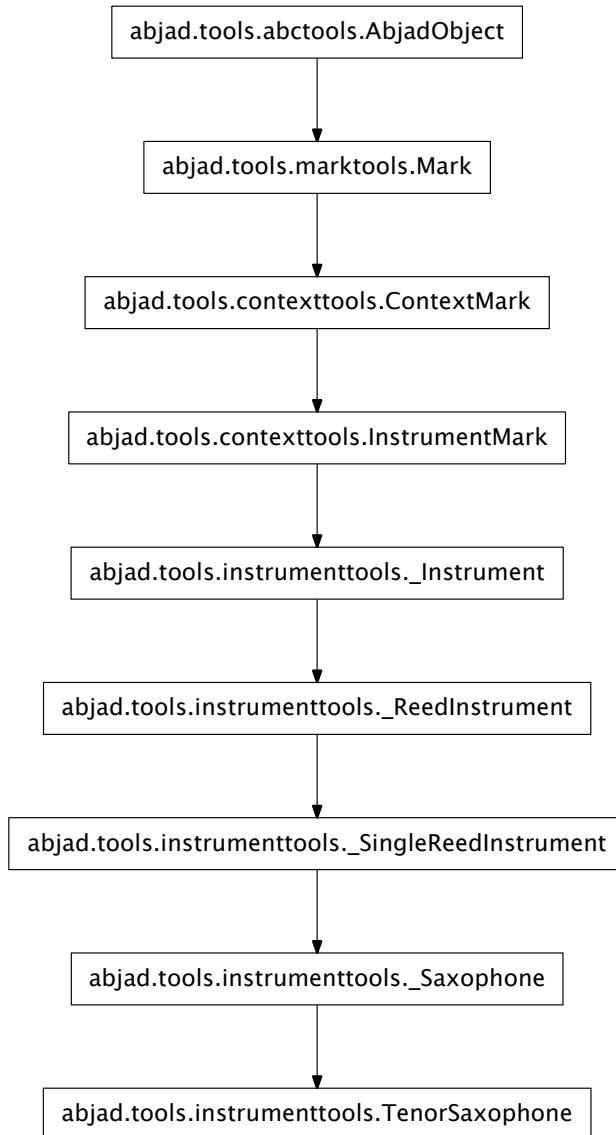
SopranoVoice.__ne__(arg)

Inherited from `marktools.Mark`

SopranoVoice.__repr__()

Inherited from `marktools.Mark`

instrumenttools.TenorSaxophone



```

class instrumenttools.TenorSaxophone(**kwargs)
    New in version 2.6. Abjad model of the tenor saxophone:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.TenorSaxophone()(staff)
    TenorSaxophone()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Tenor saxophone }
  \set Staff.shortInstrumentName = \markup { Ten. sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The tenor saxophone is pitched in B-flat.

The tenor saxophone targets staff context by default.

Read-only Properties

`TenorSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`TenorSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

`TenorSaxophone.is_primary_instrument`

Inherited from `instrumenttools._Instrument`

`TenorSaxophone.is_secondary_instrument`

Inherited from `instrumenttools._Instrument`

`TenorSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

`TenorSaxophone.lilypond_format`

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`TenorSaxophone.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`TenorSaxophone.target_context`

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`TenorSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

`TenorSaxophone.all_clefs`

Inherited from `instrumenttools._Instrument`

`TenorSaxophone.instrument_name`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```


Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.pitch_range`

Inherited from `instrumenttools._Instrument`

`TenorSaxophone.primary_clefs`

Inherited from `instrumenttools._Instrument`

`TenorSaxophone.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.sounding_pitch_of_fingered_middle_c`

Inherited from `instrumenttools._Instrument`

Methods

`TenorSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`TenorSaxophone.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`TenorSaxophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`TenorSaxophone.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`TenorSaxophone.__call__(*args)`

Inherited from `marktools.Mark`

`TenorSaxophone.__delattr__(*args)`

Inherited from `marktools.Mark`

`TenorSaxophone.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

TenorSaxophone.**__gt__**(arg)
Abjad objects by default do not implement this method.
Raise exception
Inherited from abctools.AbjadObject

TenorSaxophone.**__hash__**()
Inherited from contexttools.InstrumentMark

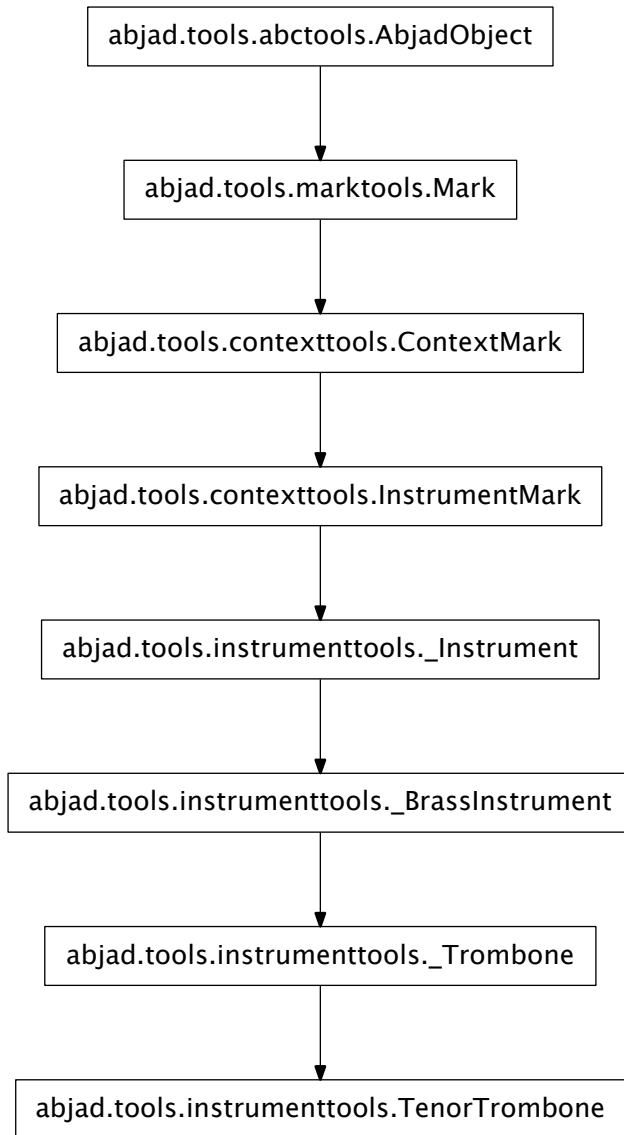
TenorSaxophone.**__le__**(arg)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from abctools.AbjadObject

TenorSaxophone.**__lt__**(arg)
Abjad objects by default do not implement this method.
Raise exception.
Inherited from abctools.AbjadObject

TenorSaxophone.**__ne__**(arg)
Inherited from marktools.Mark

TenorSaxophone.**__repr__**()
Inherited from marktools.Mark

instrumenttools.TenorTrombone



```

class instrumenttools.TenorTrombone(**kwargs)
    New in version 2.0. Abjad model of the tenor trombone:

    >>> staff = Staff("c'8 d'8 e'8 f'8")
    >>> contexttools.ClefMark('bass')(staff)
    ClefMark('bass')(Staff{4})

    >>> instrumenttools.TenorTrombone()(staff)
    TenorTrombone()(Staff{4})
  
```

```
>>> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Tenor trombone }
  \set Staff.shortInstrumentName = \markup { Ten. trb. }
  c'8
  d'8
  e'8
  f'8
}
```

The tenor trombone targets staff context by default.

Read-only Properties

TenorTrombone.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

TenorTrombone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

TenorTrombone.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

TenorTrombone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

TenorTrombone.is_primary_instrument

Inherited from `instrumenttools._Instrument`

TenorTrombone.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

TenorTrombone.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

TenorTrombone.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

TenorTrombone.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

TenorTrombone.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

TenorTrombone.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

TenorTrombone.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

TenorTrombone.all_clefs

Inherited from `instrumenttools._Instrument`

TenorTrombone.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`TenorTrombone.instrument_name_markup`

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

`TenorTrombone.pitch_range`

Inherited from `instrumenttools._Instrument`

`TenorTrombone.primary_clefs`

Inherited from `instrumenttools._Instrument`

`TenorTrombone.short_instrument_name`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from `contexttools.InstrumentMark`

`TenorTrombone.short_instrument_name_markup`

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from `contexttools.InstrumentMark`

TenorTrombone.**sounding_pitch_of_fingered_middle_c**

Inherited from `instrumenttools._Instrument`

Methods

TenorTrombone.**attach** (*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

TenorTrombone.**detach** ()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

TenorTrombone.**get_default_performer_name** (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

TenorTrombone.**get_performer_names** ()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

TenorTrombone.**__call__** (*args)

Inherited from `marktools.Mark`

TenorTrombone.**__delattr__** (*args)

Inherited from `marktools.Mark`

TenorTrombone.**__eq__** (arg)

Inherited from `contexttools.InstrumentMark`

TenorTrombone.**__ge__** (arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

TenorTrombone.**__gt__** (arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`


```
TenorTrombone.__hash__()
    Inherited from contexttools.InstrumentMark

TenorTrombone.__le__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject

TenorTrombone.__lt__(arg)
    Abjad objects by default do not implement this method.

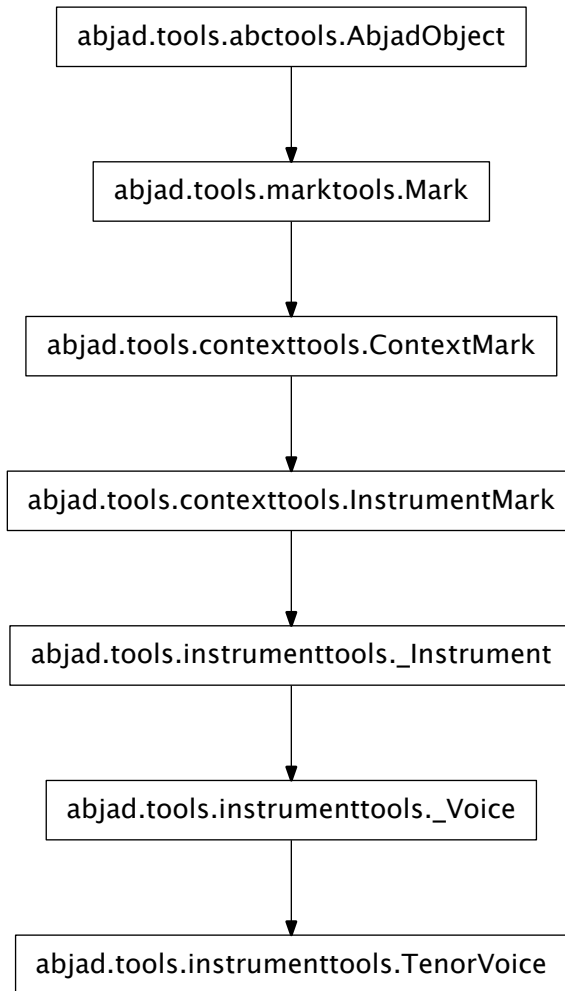
    Raise exception.

    Inherited from abctools.AbjadObject

TenorTrombone.__ne__(arg)
    Inherited from marktools.Mark

TenorTrombone.__repr__()
    Inherited from marktools.Mark
```

instrumenttools.TenorVoice



```

class instrumenttools.TenorVoice(**kwargs)
    New in version 2.8. Abjad model of the tenor voice:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.TenorVoice()(staff)
    TenorVoice()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Tenor voice }
      \set Staff.shortInstrumentName = \markup { Tenor }
      c'8
      d'8
  
```

```
e' 8
f' 8
}
```

The tenor voice targets staff context by default.

Read-only Properties

TenorVoice.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

TenorVoice.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

TenorVoice.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

TenorVoice.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

TenorVoice.is_primary_instrument

Inherited from `instrumenttools._Instrument`

TenorVoice.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

TenorVoice.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

TenorVoice.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

TenorVoice.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

TenorVoice.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

TenorVoice.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

TenorVoice.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

TenorVoice.all_clefs

Inherited from `instrumenttools._Instrument`

TenorVoice.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

TenorVoice.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

TenorVoice.pitch_range

Inherited from instrumenttools._Instrument

TenorVoice.primary_clefs

Inherited from instrumenttools._Instrument

TenorVoice.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

TenorVoice.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

TenorVoice.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

TenorVoice.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

TenorVoice.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

TenorVoice.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

TenorVoice.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

TenorVoice.**__call__**(*args)

Inherited from `marktools.Mark`

TenorVoice.**__delattr__**(*args)

Inherited from `marktools.Mark`

TenorVoice.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

TenorVoice.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

TenorVoice.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

TenorVoice.**__hash__**()

Inherited from `contexttools.InstrumentMark`

TenorVoice.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

TenorVoice.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

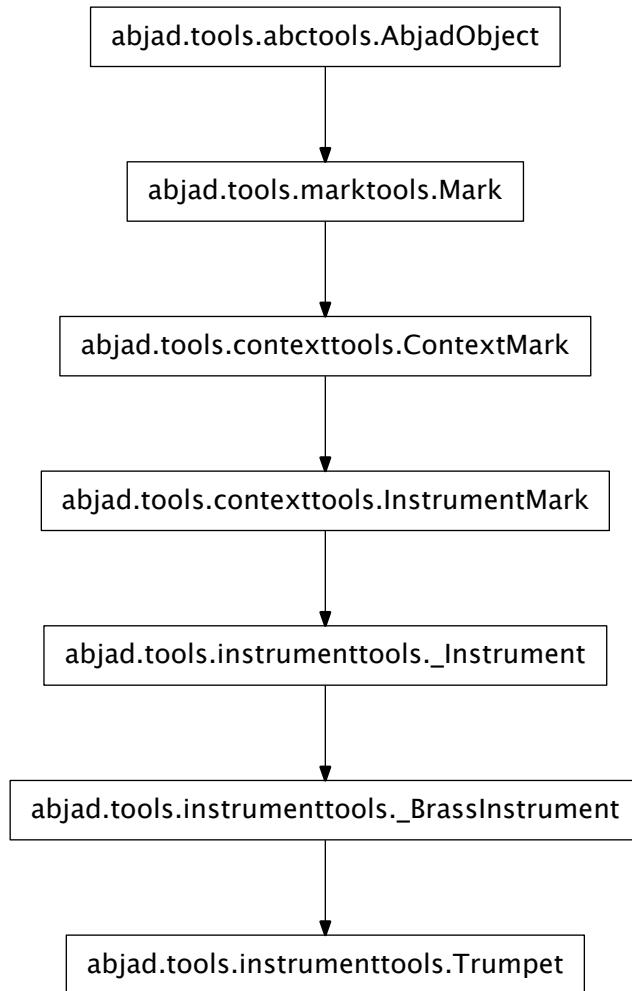
TenorVoice.__ne__(arg)

Inherited from `marktools.Mark`

TenorVoice.__repr__()

Inherited from `marktools.Mark`

instrumenttools.Trumpet



```

class instrumenttools.Trumpet (**kwargs)
    New in version 2.0. Abjad model of the trumpet:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.Trumpet() (staff)
    Trumpet() (Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Trumpet }
      \set Staff.shortInstrumentName = \markup { Tp. }
      c'8
      d'8
  
```



```
e' 8
f' 8
}
```

The trumpet targets staff context by default.

Read-only Properties

Trumpet.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Trumpet.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Trumpet.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Trumpet.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Trumpet.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Trumpet.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Trumpet.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Trumpet.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\\set Staff.instrumentName = \\markup { Flute }',
 '\\set Staff.shortInstrumentName = \\markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Trumpet.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Trumpet.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Trumpet.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Trumpet.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Trumpet.all_clefs

Inherited from `instrumenttools._Instrument`

Trumpet.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Trumpet.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Trumpet.pitch_range

Inherited from instrumenttools._Instrument

Trumpet.primary_clefs

Inherited from instrumenttools._Instrument

Trumpet.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Trumpet.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Trumpet.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

Trumpet.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Trumpet.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Trumpet.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Trumpet.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Trumpet.**__call__**(*args)

Inherited from `marktools.Mark`

Trumpet.**__delattr__**(*args)

Inherited from `marktools.Mark`

Trumpet.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

Trumpet.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Trumpet.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Trumpet.**__hash__**()

Inherited from `contexttools.InstrumentMark`

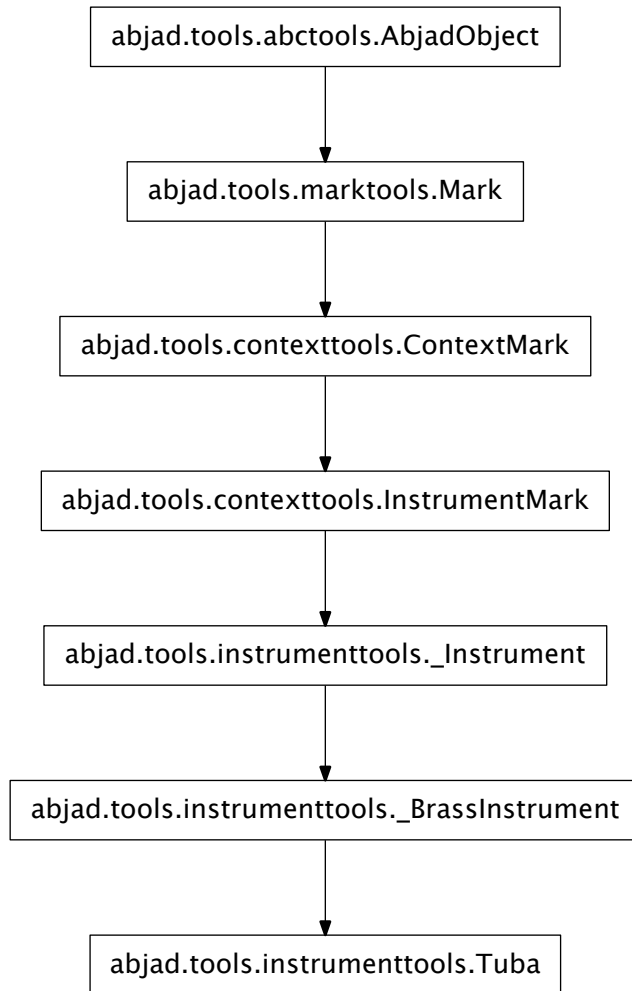
Trumpet.**__le__**(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

Trumpet.**__lt__**(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

Trumpet.**__ne__**(arg)
 Inherited from `marktools.Mark`

Trumpet.**__repr__**()
 Inherited from `marktools.Mark`

instrumenttools.Tuba



class instrumenttools.**Tuba** (**kwargs)

New in version 2.0. Abjad model of the tuba:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

>>> instrumenttools.Tuba()(staff)
Tuba()(Staff{4})

>>> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Tuba }

```

```

\set Staff.shortInstrumentName = \markup { Tb. }
c' 8
d' 8
e' 8
f' 8
}

```

The tuba targets staff context by default.

Read-only Properties

Tuba.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Tuba.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Tuba.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Tuba.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Tuba.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Tuba.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Tuba.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Tuba.lilypond_format

Read-only LilyPond input format of instrument mark:

```

>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Inherited from `contexttools.InstrumentMark`

Tuba.**start_component**

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Tuba.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Tuba.**target_context**

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Tuba.**traditional_pitch_range**

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Tuba.**all_clefs**

Inherited from `instrumenttools._Instrument`

Tuba.**instrument_name**

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Tuba.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Tuba.pitch_range

Inherited from instrumenttools._Instrument

Tuba.primary_clefs

Inherited from instrumenttools._Instrument

Tuba.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Tuba.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Tuba.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

Tuba.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Tuba.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Tuba.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Tuba.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Tuba.**__call__**(*args)

Inherited from `marktools.Mark`

Tuba.**__delattr__**(*args)

Inherited from `marktools.Mark`

Tuba.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

Tuba.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Tuba.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Tuba.**__hash__**()

Inherited from `contexttools.InstrumentMark`

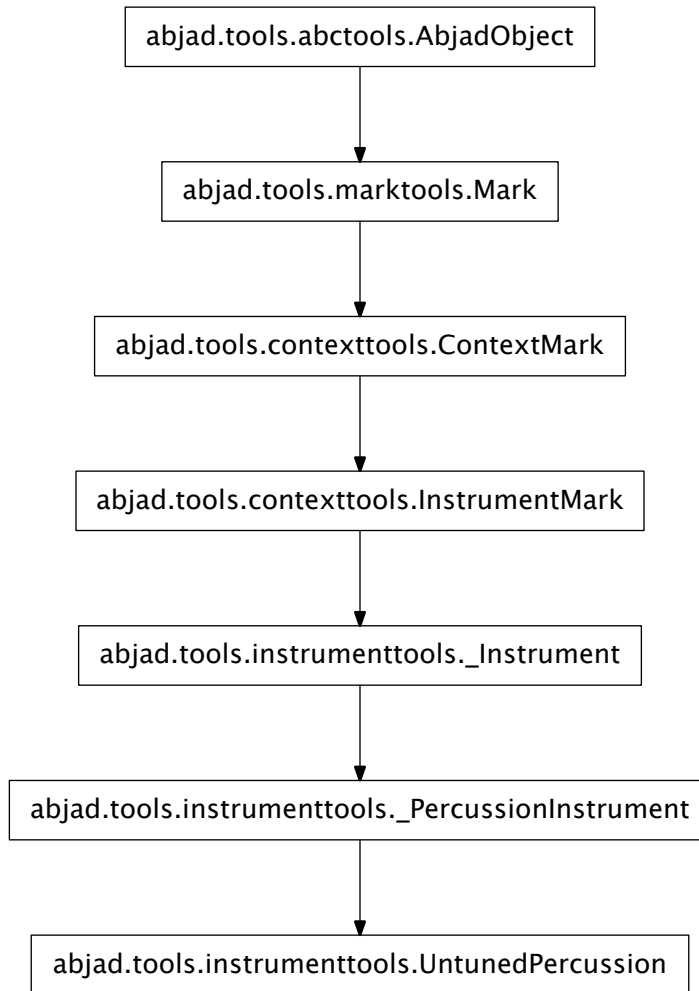
Tuba.**__le__**(*arg*)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

Tuba.**__lt__**(*arg*)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

Tuba.**__ne__**(*arg*)
 Inherited from `marktools.Mark`

Tuba.**__repr__**()
 Inherited from `marktools.Mark`

instrumenttools.UntunedPercussion



class instrumenttools.UntunedPercussion(**kwargs)

New in version 2.0. Abjad model of untuned percussion:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> instrumenttools.UntunedPercussion()(staff)
UntunedPercussion()(Staff{4})
```

```
>>> f(staff)
```

```
\new Staff {
  \set Staff.instrumentName = \markup { Untuned percussion }
  \set Staff.shortInstrumentName = \markup { Perc. }
  c'8
  d'8
```

```
e' 8
f' 8
}
```

Untuned percussion targets the staff context by default.

Read-only Properties

`UntunedPercussion.default_instrument_name`

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`UntunedPercussion.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

`UntunedPercussion.effective_context`

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

`UntunedPercussion.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

`UntunedPercussion.is_primary_instrument`

Inherited from `instrumenttools._Instrument`

`UntunedPercussion.is_secondary_instrument`

Inherited from `instrumenttools._Instrument`

`UntunedPercussion.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

`UntunedPercussion.lilypond_format`

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

UntunedPercussion.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

UntunedPercussion.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

UntunedPercussion.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

UntunedPercussion.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

UntunedPercussion.all_clefs

Inherited from `instrumenttools._Instrument`

UntunedPercussion.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

UntunedPercussion.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

UntunedPercussion.pitch_range

Inherited from instrumenttools._Instrument

UntunedPercussion.primary_clefs

Inherited from instrumenttools._Instrument

UntunedPercussion.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

UntunedPercussion.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

UntunedPercussion.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

UntunedPercussion.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from contexttools.ContextMark

UntunedPercussion.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from contexttools.ContextMark

UntunedPercussion.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from instrumenttools._Instrument

UntunedPercussion.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from instrumenttools._Instrument

Special Methods

UntunedPercussion.**__call__**(*args)

Inherited from marktools.Mark

UntunedPercussion.**__delattr__**(*args)

Inherited from marktools.Mark

UntunedPercussion.**__eq__**(arg)

Inherited from contexttools.InstrumentMark

UntunedPercussion.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

UntunedPercussion.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

UntunedPercussion.**__hash__**()

Inherited from contexttools.InstrumentMark

UntunedPercussion.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

UntunedPercussion.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

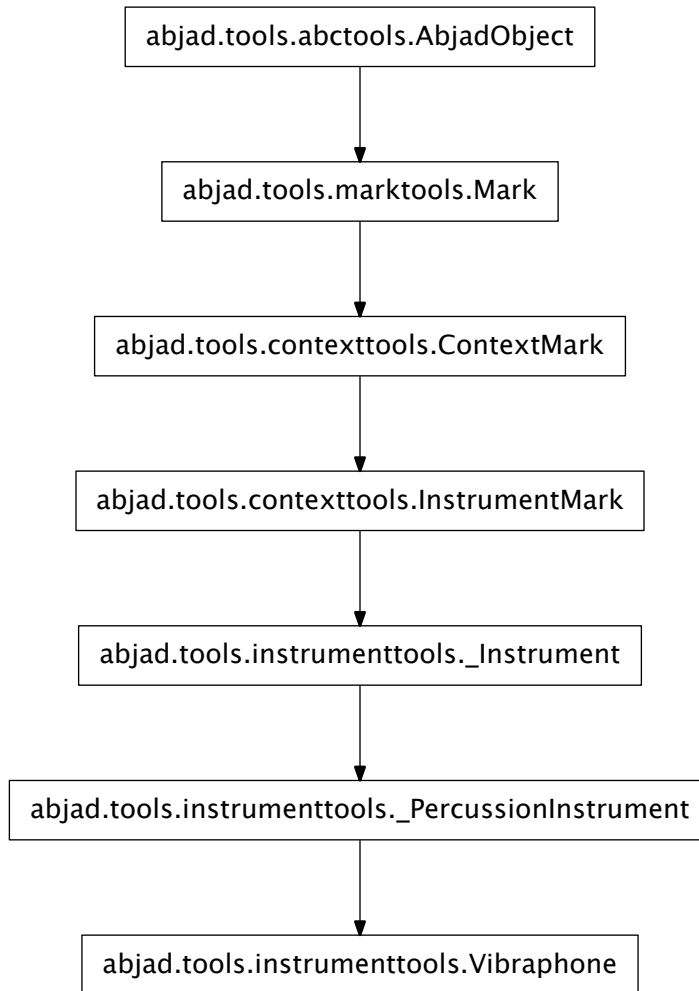
UntunedPercussion.__ne__(arg)

Inherited from `marktools.Mark`

UntunedPercussion.__repr__()

Inherited from `marktools.Mark`

instrumenttools.Vibraphone



```

class instrumenttools.Vibraphone(**kwargs)
    New in version 2.0. Abjad model of the vibraphone:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.Vibraphone()(staff)
    Vibraphone()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Vibraphone }
      \set Staff.shortInstrumentName = \markup { Vibr. }
      c'8
      d'8
  
```

```
e' 8
f' 8
}
```

The vibraphone targets staff context by default.

Read-only Properties

Vibraphone.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Vibraphone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Vibraphone.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Vibraphone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Vibraphone.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Vibraphone.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Vibraphone.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Vibraphone.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Vibraphone.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Vibraphone.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Vibraphone.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Vibraphone.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Vibraphone.all_clefs

Inherited from `instrumenttools._Instrument`

Vibraphone.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Vibraphone.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Vibraphone.pitch_range

Inherited from instrumenttools._Instrument

Vibraphone.primary_clefs

Inherited from instrumenttools._Instrument

Vibraphone.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Vibraphone.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Vibraphone.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

Vibraphone.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from contexttools.ContextMark

Vibraphone.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from contexttools.ContextMark

Vibraphone.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from instrumenttools._Instrument

Vibraphone.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from instrumenttools._Instrument

Special Methods

Vibraphone.**__call__**(*args)

Inherited from marktools.Mark

Vibraphone.**__delattr__**(*args)

Inherited from marktools.Mark

Vibraphone.**__eq__**(arg)

Inherited from contexttools.InstrumentMark

Vibraphone.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

Vibraphone.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

Vibraphone.**__hash__**()

Inherited from contexttools.InstrumentMark

`Vibraphone.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Vibraphone.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

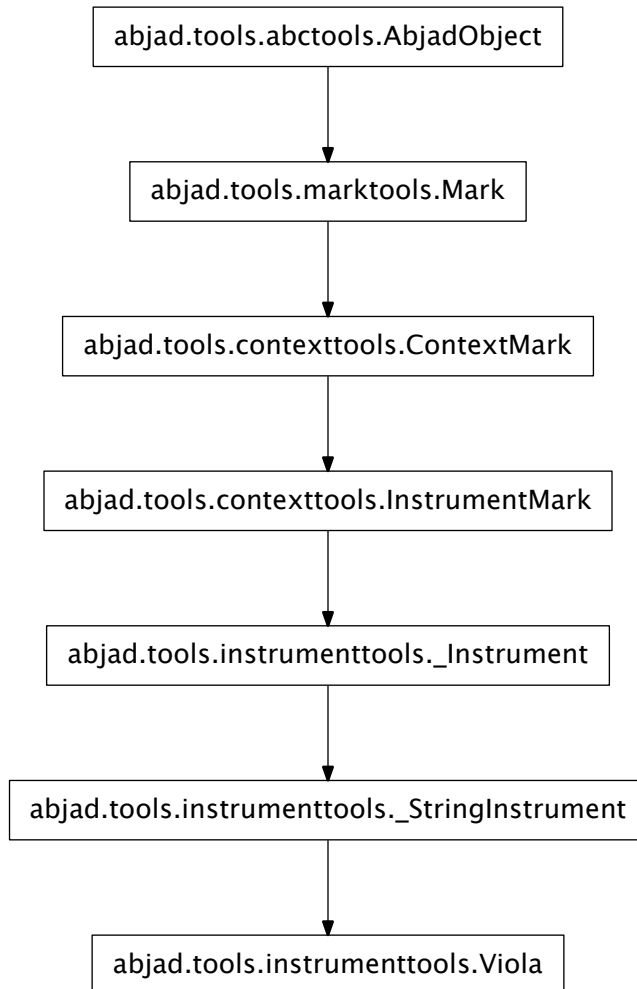
`Vibraphone.__ne__(arg)`

Inherited from `marktools.Mark`

`Vibraphone.__repr__()`

Inherited from `marktools.Mark`

instrumenttools.Viola



class instrumenttools.**Viola** (**kwargs)

New in version 2.0. Abjad model of the viola:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('alto')(staff)
ClefMark('alto')(Staff{4})

>>> instrumenttools.Viola()(staff)
Viola()(Staff{4})

>>> f(staff)
\new Staff {
  \clef "alto"
  \set Staff.instrumentName = \markup { Viola }

```



```

\set Staff.shortInstrumentName = \markup { Va. }
c'8
d'8
e'8
f'8
}

```

The viola targets staff context by default.

Read-only Properties

Viola.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Viola.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Viola.effective_context

Read-only reference to effective context of context mark:

```

>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True

```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Viola.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Viola.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Viola.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Viola.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Viola.lilypond_format

Read-only LilyPond input format of instrument mark:

```

>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Inherited from `contexttools.InstrumentMark`

Viola.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Viola.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Viola.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Viola.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Viola.all_clefs

Inherited from `instrumenttools._Instrument`

Viola.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Viola.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Viola.pitch_range

Inherited from instrumenttools._Instrument

Viola.primary_clefs

Inherited from instrumenttools._Instrument

Viola.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Viola.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Viola.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

Viola.**attach**(*start_component*)

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

Viola.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

Viola.**get_default_performer_name**(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

Viola.**get_performer_names**()

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

Viola.**__call__**(*args)

Inherited from `marktools.Mark`

Viola.**__delattr__**(*args)

Inherited from `marktools.Mark`

Viola.**__eq__**(arg)

Inherited from `contexttools.InstrumentMark`

Viola.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Viola.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Viola.**__hash__**()

Inherited from `contexttools.InstrumentMark`

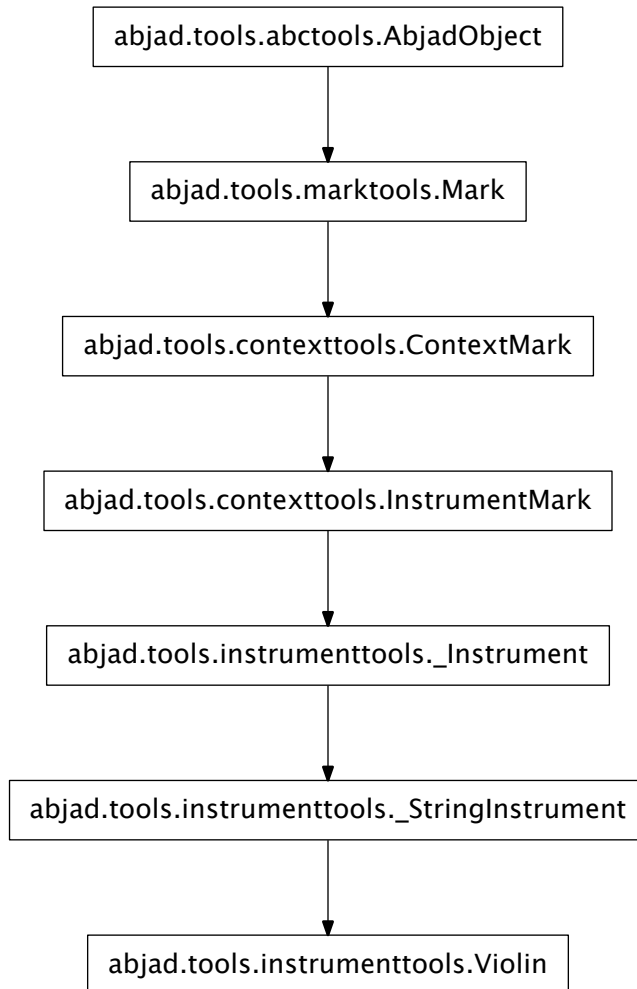
Viola.**__le__**(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

Viola.**__lt__**(arg)
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

Viola.**__ne__**(arg)
 Inherited from `marktools.Mark`

Viola.**__repr__**()
 Inherited from `marktools.Mark`

instrumenttools.Violin



```

class instrumenttools.Violin(**kwargs)
    New in version 2.0. Abjad model of the violin:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.Violin()(staff)
    Violin()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Violin }
      \set Staff.shortInstrumentName = \markup { Vn. }
      c'8
      d'8
  
```

```
e'8
f'8
}
```

The violin targets staff context by default.

Read-only Properties

Violin.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Violin.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Violin.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Violin.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Violin.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Violin.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Violin.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Violin.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\set Staff.instrumentName = \markup { Flute }',
 '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Violin.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Violin.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Violin.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Violin.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Violin.all_clefs

Inherited from `instrumenttools._Instrument`

Violin.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Violin.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Violin.pitch_range

Inherited from instrumenttools._Instrument

Violin.primary_clefs

Inherited from instrumenttools._Instrument

Violin.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Violin.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Violin.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

`Violin.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`Violin.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.start_component
Note("c'4")

>>> context_mark.detach()
ContextMark()

>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`Violin.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`Violin.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`Violin.__call__(*args)`

Inherited from `marktools.Mark`

`Violin.__delattr__(*args)`

Inherited from `marktools.Mark`

`Violin.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`Violin.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Violin.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Violin.__hash__()`

Inherited from `contexttools.InstrumentMark`

Violin.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Violin.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

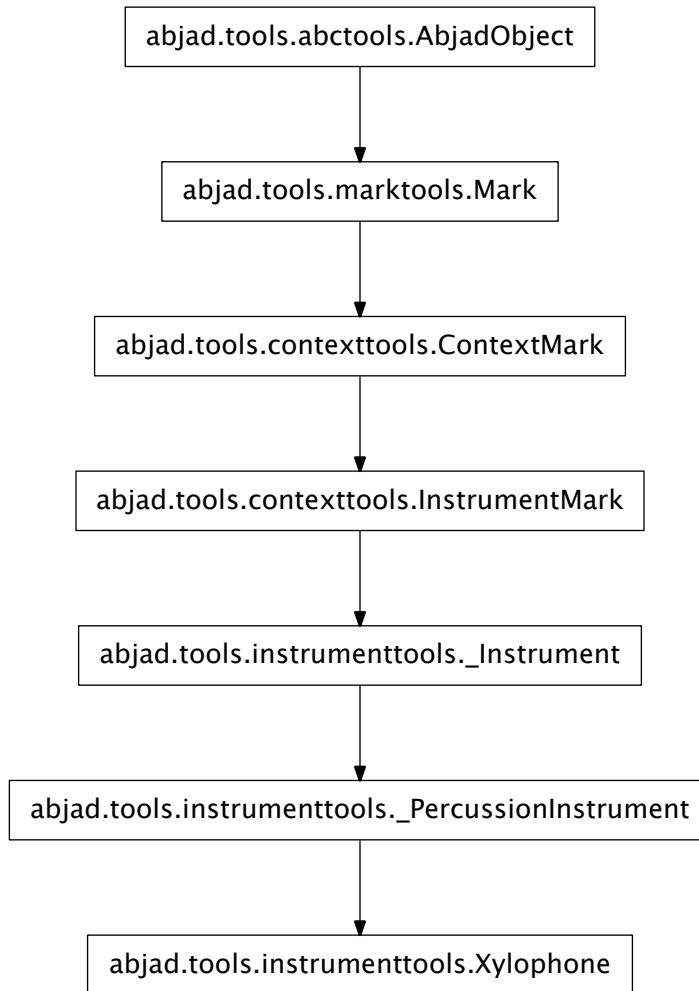
Violin.__ne__(arg)

Inherited from `marktools.Mark`

Violin.__repr__()

Inherited from `marktools.Mark`

instrumenttools.Xylophone



```

class instrumenttools.Xylophone(**kwargs)
    New in version 2.0. Abjad model of the xylphone:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> instrumenttools.Xylophone()(staff)
    Xylophone()(Staff{4})

    >>> f(staff)
    \new Staff {
      \set Staff.instrumentName = \markup { Xylophone }
      \set Staff.shortInstrumentName = \markup { Xyl. }
      c'8
      d'8
  
```

```
e' 8
f' 8
}
```

The xylophone targets staff context by default.

Read-only Properties

Xylophone.default_instrument_name

Read-only default instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Xylophone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Inherited from `contexttools.InstrumentMark`

Xylophone.effective_context

Read-only reference to effective context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.effective_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Xylophone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Inherited from `instrumenttools._Instrument`

Xylophone.is_primary_instrument

Inherited from `instrumenttools._Instrument`

Xylophone.is_secondary_instrument

Inherited from `instrumenttools._Instrument`

Xylophone.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Inherited from `instrumenttools._Instrument`

Xylophone.lilypond_format

Read-only LilyPond input format of instrument mark:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.lilypond_format
['\\set Staff.instrumentName = \\markup { Flute }',
 '\\set Staff.shortInstrumentName = \\markup { Fl. }']
```

Return list.

Inherited from `contexttools.InstrumentMark`

Xylophone.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Xylophone.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Xylophone.target_context

Read-only reference to target context of context mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)

>>> context_mark.target_context is None
True
```

Return context mark or none.

Inherited from `contexttools.ContextMark`

Xylophone.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Inherited from `instrumenttools._Instrument`

Read/write Properties

Xylophone.all_clefs

Inherited from `instrumenttools._Instrument`

Xylophone.instrument_name

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name
'Flute'
```

Set instrument name:

```
>>> instrument.instrument_name = 'Alto Flute'
>>> instrument.instrument_name
'Alto Flute'
```

Return string.

Inherited from `contexttools.InstrumentMark`

Xylophone.instrument_name_markup

Get instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.instrument_name_markup
Markup(('Flute',))
```

Set instrument name:

```
>>> instrument.instrument_name_markup = 'Alto Flute'
>>> instrument.instrument_name_markup
Markup(('Alto Flute',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Xylophone.pitch_range

Inherited from instrumenttools._Instrument

Xylophone.primary_clefs

Inherited from instrumenttools._Instrument

Xylophone.short_instrument_name

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
>>> instrument.short_instrument_name = 'Alto Fl.'
>>> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Inherited from contexttools.InstrumentMark

Xylophone.short_instrument_name_markup

Get short instrument name:

```
>>> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
>>> instrument.short_instrument_name_markup
Markup(('Fl.',))
```

Set short instrument name:

```
>>> instrument.short_instrument_name_markup = 'Alto Fl.'
>>> instrument.short_instrument_name_markup
Markup(('Alto Fl.',))
```

Return markup.

Inherited from contexttools.InstrumentMark

Xylophone.sounding_pitch_of_fingered_middle_c

Inherited from instrumenttools._Instrument

Methods

`Xylophone.attach(start_component)`

Make sure no context mark of same type is already attached to score component that starts with start component.

Inherited from `contexttools.ContextMark`

`Xylophone.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> context_mark = contexttools.ContextMark()(note)
```

```
>>> context_mark.start_component
Note("c'4")
```

```
>>> context_mark.detach()
ContextMark()
```

```
>>> context_mark.start_component is None
True
```

Return context mark.

Inherited from `contexttools.ContextMark`

`Xylophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Inherited from `instrumenttools._Instrument`

`Xylophone.get_performer_names()`

New in version 2.5. Get performer names.

Inherited from `instrumenttools._Instrument`

Special Methods

`Xylophone.__call__(*args)`

Inherited from `marktools.Mark`

`Xylophone.__delattr__(*args)`

Inherited from `marktools.Mark`

`Xylophone.__eq__(arg)`

Inherited from `contexttools.InstrumentMark`

`Xylophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Xylophone.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Xylophone.__hash__()`

Inherited from `contexttools.InstrumentMark`

`Xylophone.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Xylophone.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Xylophone.__ne__(arg)`
 Inherited from `marktools.Mark`

`Xylophone.__repr__()`
 Inherited from `marktools.Mark`

functions

`instrumenttools.default_instrument_name_to_instrument_class`

`instrumenttools.default_instrument_name_to_instrument_class(default_instrument_name)`
 New in version 2.5. Change *default_instrument_name* to class name:

```
>>> instrumenttools.default_instrument_name_to_instrument_class('clarinet in E-flat')
<class 'abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet'>
```

Return class.

When *default_instrument_name* matches no instrument class:

```
>>> instrumenttools.default_instrument_name_to_instrument_class('foo') is None
True
```

Return none.

`instrumenttools.iterate_notes_and_chords_in_expr_outside_traditional_instrument_ranges`

`instrumenttools.iterate_notes_and_chords_in_expr_outside_traditional_instrument_ranges(expr)`
 New in version 2.0. Iterate notes and chords in *expr* outside traditional instrument ranges:

```
>>> staff = Staff("c'8 r8 <d fs>8 r8")
>>> instrumenttools.Violin()(staff)
Violin()(Staff{4})

>>> list(
... instrumenttools.iterate_notes_and_chords_in_expr_outside_traditional_instrument_ranges(
... staff))
[Chord('<d fs>8')]
```

Return generator.

instrumenttools.list_instrument_names

`instrumenttools.list_instrument_names()`

New in version 2.5. List instrument names:

```
>>> for instrument_name in instrumenttools.list_instrument_names():
...     instrument_name
...
'accordion'
'alto flute'
'alto saxophone'
'alto trombone'
'clarinet in B-flat'
'baritone saxophone'
'baritone voice'
'bass clarinet'
'bass flute'
'bass saxophone'
'bass trombone'
'bass voice'
'bassoon'
'cello'
'clarinet in A'
'contrabass'
'contrabass clarinet'
'contrabass flute'
'contrabass saxophone'
'contrabassoon'
'contralto voice'
'clarinet in E-flat'
'English horn'
'flute'
'horn'
'glockenspiel'
'guitar'
'harp'
'harpsichord'
'marimba'
'mezzo-soprano voice'
'oboe'
'piano'
'piccolo'
'sopranino saxophone'
'soprano saxophone'
'soprano voice'
'tenor saxophone'
'tenor trombone'
'tenor voice'
'trumpet'
'tuba'
'untuned percussion'
'vibraphone'
'viola'
'violin'
'xylophone'
```

Return list.

instrumenttools.list_instruments

`instrumenttools.list_instruments` (*klassen=None*)

New in version 2.5. List instruments in instrumenttools module:

```
>>> for instrument in instrumenttools.list_instruments()[:5]:
...     instrument
...
<class 'abjad.tools.instrumenttools.Accordion.Accordion.Accordion'>
<class 'abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute'>
<class 'abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone'>
<class 'abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone'>
<class 'abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet'>
```

Return list.

instrumenttools.list_primary_instruments

`instrumenttools.list_primary_instruments` ()

New in version 2.5. List primary instruments:

```
>>> for primary_instrument in instrumenttools.list_primary_instruments():
...     primary_instrument
...
<class 'abjad.tools.instrumenttools.Accordion.Accordion.Accordion'>
<class 'abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone'>
<class 'abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet'>
<class 'abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice'>
<class 'abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice'>
<class 'abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon'>
<class 'abjad.tools.instrumenttools.Cello.Cello.Cello'>
<class 'abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass'>
<class 'abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice'>
<class 'abjad.tools.instrumenttools.Flute.Flute.Flute'>
<class 'abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn'>
<class 'abjad.tools.instrumenttools.Guitar.Guitar.Guitar'>
<class 'abjad.tools.instrumenttools.Harp.Harp.Harp'>
<class 'abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord'>
<class 'abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice'>
<class 'abjad.tools.instrumenttools.Oboe.Oboe.Oboe'>
<class 'abjad.tools.instrumenttools.Piano.Piano.Piano'>
<class 'abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice'>
<class 'abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone'>
<class 'abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice'>
<class 'abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet'>
<class 'abjad.tools.instrumenttools.Tuba.Tuba.Tuba'>
<class 'abjad.tools.instrumenttools.Viola.Viola.Viola'>
<class 'abjad.tools.instrumenttools.Violin.Violin.Violin'>
```

Return list

instrumenttools.list_secondary_instruments

`instrumenttools.list_secondary_instruments` ()

New in version 2.5. List secondary instruments:

```
>>> for secondary_instrument in instrumenttools.list_secondary_instruments()[5]:
...     secondary_instrument
...
<class 'abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute'>
<class 'abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone'>
<class 'abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone'>
<class 'abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet'>
<class 'abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute'>
```

Return list

instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs

`instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs`(*expr*, *percussion_clef_is_allowed=True*)

New in version 2.0. True when notes and chords in *expr* are on expected clefs:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})
>>> instrumenttools.Violin()(staff)
Violin()(Staff{4})

>>> instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs(staff)
True
```

False otherwise:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('alto')(staff)
ClefMark('alto')(Staff{4})
>>> instrumenttools.Violin()(staff)
Violin()(Staff{4})

>>> instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs(staff)
False
```

Allow percussion clef when *percussion_clef_is_allowed* is true:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.ClefMark('percussion')(staff)
ClefMark('percussion')(Staff{4})
>>> instrumenttools.Violin()(staff)
Violin()(Staff{4})

>>> f(staff)
\new Staff {
  \clef "percussion"
  \set Staff.instrumentName = \markup { Violin }
  \set Staff.shortInstrumentName = \markup { Vn. }
  c'8
  d'8
  e'8
  f'8
}
```

```
>>> instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs(
...     staff, percussion_clef_is_allowed=True)
True
```

Disallow percussion clef when *percussion_clef_is_allowed* is false:

```
>>> instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs(
...     staff, percussion_clef_is_allowed=False)
False
```

Return boolean.

instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges

`instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges(expr)`
 New in version 2.0. True when notes and chords in *expr* are within traditional instrument ranges:

```
>>> staff = Staff("c'8 r8 <d' fs'>8 r8")
>>> instrumenttools.Violin() (staff)
Violin() (Staff{4})

>>> instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges(
...     staff)
True
```

False otherwise:

```
>>> staff = Staff("c'8 r8 <d fs>8 r8")
>>> instrumenttools.Violin() (staff)
Violin() (Staff{4})

>>> instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges(
...     staff)
False
```

Return boolean.

instrumenttools.transpose_from_fingered_pitch_to_sounding_pitch

`instrumenttools.transpose_from_fingered_pitch_to_sounding_pitch(expr)`
 New in version 2.0. Transpose notes and chords in *expr* from sounding pitch to fingered pitch:

```
>>> staff = Staff("<c' e' g'>4 d'4 r4 e'4")
>>> instrumenttools.BFlatClarinet() (staff)
BFlatClarinet() (Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  <c' e' g'>4
  d'4
  r4
  e'4
}
```

```
>>> for leaf in staff.leaves:
...     leaf.written_pitch_indication_is_at_sounding_pitch = False

>>> instrumenttools.transpose_from_fingered_pitch_to_sounding_pitch(staff)

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  <bf d' f'>4
  c'4
  r4
  d'4
}
```

Return none.

instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch

instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch(*expr*)

New in version 2.0. Transpose notes and chords in *expr* from sounding pitch to fingered pitch:

```
>>> staff = Staff("<c' e' g'>4 d'4 r4 e'4")
>>> instrumenttools.BFlatClarinet()(staff)
BFlatClarinet()(Staff{4})

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  <c' e' g'>4
  d'4
  r4
  e'4
}

>>> instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch(staff)

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  <d' fs' a'>4
  e'4
  r4
  fs'4
}
```

Return none.

iotools

functions

iotools.clear_terminal

`iotools.clear_terminal()`

New in version 2.0. Run `clear` if OS is POSIX-compliant (UNIX / Linux / MacOS).

Run `cls` if OS is not POSIX-compliant (Windows):

```
>>> iotools.clear_terminal()
```

Return `none`.

iotools.f

`iotools.f(expr)`

Format *expr* and print to standard out:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> f(staff)
```

```
\new Staff {  
  c'8  
  d'8  
  e'8  
  f'8  
}
```

Return `none`.

iotools.get_last_output_file_name

`iotools.get_last_output_file_name()`

Get last output file name like `6222.ly`.

Return string.

iotools.get_next_output_file_name

`iotools.get_next_output_file_name()`

Get next output file name like `6223.ly`.

Return string.

iotools.log

`iotools.log()`

Open the LilyPond log file in operating system-specific text editor:

```
>>> iotools.log()
```

```
GNU LilyPond 2.12.2
Processing `0440.ly'
Parsing...
Interpreting music...
Preprocessing graphical objects...
Finding the ideal number of pages...
Fitting music on 1 page...
Drawing systems...
Layout output to `0440.ps'...
Converting to `./0440.pdf'...
```

Exit text editor in the usual way.

Return none.

iotools.ly

`iotools.ly` (*target=-1*)

Open the last LilyPond output file in text editor:

```
>>> iotools.ly()

% Abjad revision 2162
% 2009-05-31 14:29

\version "2.12.2"
\include "english.ly"
\include "/Path/to/abjad/trunk/abjad/cfg/abjad.scm"

{
  c'4
}
```

Open the next-to-last LilyPond output file in text editor:

```
>>> iotools.ly(-2)
```

Exit text editor in the usual way.

Return none.

iotools.p

`iotools.p` (**args*)

New in version 2.7. Parse *args* as LilyPond string:

```
>>> p("{c'4 d'4 e'4 f'4}")
{c'4, d'4, e'4, f'4}

>>> container = _
>>> f(container)
{
  c'4
  d'4
  e'4
  f'4
}
```


A pitch-name language may also be specified.

```
>>> p("{c'8 des' e' fis'}", 'nederlands')
{c'8, df'8, e'8, fs'8}
```

Return Abjad expression.

iotools.pdf

`iotools.pdf` (*target=-1*)

Open the last PDF generated by Abjad with `iotools.pdf()`.

Open the next-to-last PDF generated by Abjad with `iotools.pdf(-2)`.

Return none.

Abjad writes PDFs to the `~/ .abjad/output` directory by default.

You may change this by setting the `abjad_output` variable in the `config.py` file.

iotools.play

`iotools.play` (*expr*)

Play *expr*:

```
>>> note = Note("c'4")

>>> iotools.play(note)
```

This input creates and opens a one-note MIDI file.

Abjad outputs MIDI files of the format `filename.mid` under Windows.

Abjad outputs MIDI files of the format `filename.midi` under other operating systems.

iotools.profile_expr

`iotools.profile_expr` (*expr*, *sort_by='cum'*, *num_lines=12*, *strip_dirs=True*)

Profile *expr*:

```
>>> iotools.profile_expr('Staff(notetools.make_repeated_notes(8))')
Tue Apr  5 20:32:40 2011    _tmp_abj_profile
```

```
2852 function calls (2829 primitive calls) in 0.006 CPU seconds
```

```
Ordered by: cumulative time
```

```
List reduced from 118 to 12 due to restriction <12>
```

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.006	0.006	<string>:1(<module>)
1	0.000	0.000	0.003	0.003	make_repeated_notes.py:5(make_repeated
1	0.001	0.001	0.003	0.003	make_notes.py:12(make_notes)
1	0.000	0.000	0.003	0.003	Staff.py:21(__init__)
1	0.000	0.000	0.003	0.003	Context.py:11(__init__)
1	0.000	0.000	0.003	0.003	Container.py:23(__init__)
1	0.000	0.000	0.003	0.003	Container.py:271(_initialize_music)
2	0.000	0.000	0.002	0.001	all_are_thread_contiguous_components.p

```

52      0.001      0.000      0.002      0.000 component_to_thread_signature.py:5 (com
1       0.000      0.000      0.002      0.002 _construct_unprolated_notes.py:4 (_cons
8       0.000      0.000      0.002      0.000 make_tied_note.py:5 (make_tied_note)
8       0.000      0.000      0.002      0.000 make_tied_leaf.py:5 (make_tied_leaf)

```

Function wraps the built-in Python `cProfile` module.

Set *expr* to any string of Abjad input.

Set *sort_by* to ‘*cum*’, ‘*time*’ or ‘*calls*’.

Set *num_lines* to any positive integer.

Set *strip_dirs* to `True` to strip directory names from output lines.

Note: This function fails on some Linux distros. Some Linux distributions do not include the Python `pstats` module.

Note: This function creates the file `_tmp_abj_profile` in the directory from which it is run.

Note: For information on reading the output of the different Python profilers, see [the Python docs](#).

Changed in version 2.0: renamed `check.profile()` to `iotools.profile_expr()`.

iotools.redo

`iotools.redo(target=-1, lily_time=10)`

Rerender the last `.ly` file created in Abjad and then show the resulting PDF:

```
>>> iotools.redo()
```

Rerender the next-to-last `.ly` file created in Abjad and then show the resulting PDF:

```
>>> iotools.redo(-2)
```

Return `none`.

iotools.save_last_ly_as

`iotools.save_last_ly_as(file_name)`

New in version 2.0. Save last `ly` file as *file_name*:

```
>>> iotools.save_last_ly_as('/project/output/example-1.ly')
```

Return `none`.

iotools.save_last_pdf_as

`iotools.save_last_pdf_as(file_name)`

New in version 2.0. Save last PDF as *file_name*:

```
>>> iotools.save_last_pdf_as('/project/output/example-1.pdf')
```

Return none.

iotools.show

`iotools.show(expr, return_timing=False, suppress_pdf=False, docs=False)`

Show *expr*:

```
>>> note = Note("c'4")
>>> show(note)
```

Show *expr* and return both Abjad and LilyPond processing time in seconds:

```
>>> staff = Staff(Note("c'4") * 200)
>>> show(note, return_timing=True)
(0, 3)
```

Wrap *expr* in a LilyPond file with settings and overrides suitable for the Abjad reference manual When *docs* is true.

Return none or timing tuple.

Abjad writes LilyPond input files to the `~/ .abjad/output` directory by default.

You may change this by setting the `abjad_output` variable in the `config.py` file.

iotools.spawn_subprocess

`iotools.spawn_subprocess(command)`

New in version 2.9. Spawn subprocess, run *command*, redirect stderr to stdout and print result:

```
>>> iotools.spawn_subprocess('echo "hello world"')
hello world
```

The function is basically a reimplementaion of the deprecated `os.system()` using Python's subprocess module.

The function provides a type of shell access from the Abjad interpreter.

Return none.

iotools.write_expr_to_ly

`iotools.write_expr_to_ly(expr, file_name, print_status=False, tagline=False, docs=False)`

Write *expr* to *file_name*:

```
>>> note = Note("c'4")
>>> iotools.write_expr_to_ly(note, '/home/user/foo.ly')
```

Return none. Changed in version 2.0: renamed `io.write_ly()` to `io.write_expr_to_ly()`.

iotools.write_expr_to_pdf

`iotools.write_expr_to_pdf(expr, file_name, print_status=False, tagline=False)`

Write *expr* to pdf *file_name*:

```
>>> note = Note("c'4")
>>> iotools.write_expr_to_pdf(note, 'one_note.pdf')
```

Return none.

iotools.z

`iotools.z(expr)`

New in version 2.8. Print the tools package-qualified indented repr of *z*.

iterationtools

functions

iterationtools.iterate_chords_in_expr

`iterationtools.iterate_chords_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.10. Iterate chords forward in *expr*:

```
>>> staff = Staff("<e' g' c''>8 a'8 r8 <d' f' b'>8 r2")
```

```
>>> f(staff)
\new Staff {
  <e' g' c''>8
  a'8
  r8
  <d' f' b'>8
  r2
}
```

```
>>> for chord in iterationtools.iterate_chords_in_expr(staff):
...     chord
Chord("<e' g' c''>8")
Chord("<d' f' b'>8")
```

Iterate chords backward in *expr*:

```
::
```

```
>>> for chord in iterationtools.iterate_chords_in_expr(staff, reverse=True):
...     chord
Chord("<d' f' b'>8")
Chord("<e' g' c''>8")
```

Ignore threads.

Return generator.

iterationtools.iterate_components_and_grace_containers_in_expr

`iterationtools.iterate_components_and_grace_containers_in_expr(expr, klass)`

Iterate components of *klass* forward in *expr*:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(voice[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> grace_notes = [Note("c'16"), Note("d'16")]
>>> gracetools.GraceContainer(grace_notes, kind='grace')(voice[1])
Note("d'8")

>>> after_grace_notes = [Note("e'16"), Note("f'16")]
>>> gracetools.GraceContainer(after_grace_notes, kind='after')(voice[1])
Note("d'8")

>>> f(voice)
\new Voice {
  c'8 [
    \grace {
      c'16
      d'16
    }
    \afterGrace
    d'8
    {
      e'16
      f'16
    }
  ]
  e'8
  f'8 ]
}

>>> x = iterationtools.iterate_components_and_grace_containers_in_expr(voice, Note)
>>> for note in x:
...     note
...
Note("c'8")
Note("c'16")
Note("d'16")
Note("d'8")
Note("e'16")
Note("f'16")
Note("e'8")
Note("f'8")
```

Include grace leaves before main leaves.

Include grace leaves after main leaves. Changed in version 2.0: renamed `iterate.grace()` to `componenttools.iterate_components_and_grace_containers_in_expr()`.

iterationtools.iterate_components_depth_first

`iterationtools.iterate_components_depth_first` (*component*, *capped=True*, *unique=True*, *forbid=None*, *direction='left'*)

New in version 1.1. Iterate components depth-first from *component*.

Todo

Add usage examples.

Changed in version 2.0: renamed `iterate.depth_first()` to `iterationtools.iterate_components_depth_first()`.

`iterationtools.iterate_components_in_expr`

`iterationtools.iterate_components_in_expr(expr, klass=None, reverse=False, start=0, stop=None)`

New in version 2.10. Iterate components forward in *expr*.

`iterationtools.iterate_containers_in_expr`

`iterationtools.iterate_containers_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.10. Iterate containers forward in *expr*:

```
>>> staff = Staff([Voice("c'8 d'8"), Voice("e'8 f'8 g'8")])
>>> Tuplet(Fraction(2, 3), staff[1][:])
Tuplet(2/3, [e'8, f'8, g'8])
>>> staff.is_parallel = True

>>> f(staff)
\new Staff <<
  \new Voice {
    c'8
    d'8
  }
  \new Voice {
    \times 2/3 {
      e'8
      f'8
      g'8
    }
  }
>>

>>> for x in iterationtools.iterate_containers_in_expr(staff):
...     x
Staff<<2>>
Voice{2}
Voice{1}
Tuplet(2/3, [e'8, f'8, g'8])
```

Iterate containers backward in *expr*:

```
>>> for x in iterationtools.iterate_containers_in_expr(staff, reverse=True):
...     x
Staff<<2>>
Voice{1}
Tuplet(2/3, [e'8, f'8, g'8])
Voice{2}
```

Ignore threads.

Return generator.

iterationtools.iterate_contexts_in_expr

`iterationtools.iterate_contexts_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.10. Iterate contexts forward in *expr*:

```
>>> staff = Staff([Voice("c'8 d'8"), Voice("e'8 f'8 g'8")])
>>> Tuplet(Fraction(2, 3), staff[1][:])
Tuplet(2/3, [e'8, f'8, g'8])
>>> staff.is_parallel = True
```

```
>>> f(staff)
\new Staff <<
  \new Voice {
    c'8
    d'8
  }
  \new Voice {
    \times 2/3 {
      e'8
      f'8
      g'8
    }
  }
>>
```

```
>>> for x in iterationtools.iterate_contexts_in_expr(staff):
...     x
Staff<<2>>
Voice{2}
Voice{1}
```

Iterate contexts backward in *expr*:

```
>>> for x in iterationtools.iterate_contexts_in_expr(staff, reverse=True):
...     x
Staff<<2>>
Voice{1}
Voice{2}
```

Ignore threads.

Return generator.

iterationtools.iterate_leaf_pairs_in_expr

`iterationtools.iterate_leaf_pairs_in_expr(expr)`

New in version 2.0. Iterate leaf pairs forward in *expr*:

```
>>> score = Score([])
>>> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'4")]
>>> score.append(Staff(notes))
>>> notes = [Note(x, (1, 4)) for x in [-12, -15, -17]]
>>> score.append(Staff(notes))
>>> contexttools.ClefMark('bass')(score[1])
ClefMark('bass')(Staff{3})
```

```
>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
    g'4
  }
  \new Staff {
    \clef "bass"
    c4
    a,4
    g,4
  }
>>

>>> for pair in iterationtools.iterate_leaf_pairs_in_expr(score):
...     pair
(Note("c'8"), Note('c4'))
(Note("c'8"), Note("d'8"))
(Note('c4'), Note("d'8"))
(Note("d'8"), Note("e'8"))
(Note("d'8"), Note('a,4'))
(Note('c4'), Note("e'8"))
(Note('c4'), Note('a,4'))
(Note("e'8"), Note('a,4'))
(Note("e'8"), Note("f'8"))
(Note('a,4'), Note("f'8"))
(Note("f'8"), Note("g'4"))
(Note("f'8"), Note('g,4'))
(Note('a,4'), Note("g'4"))
(Note('a,4'), Note('g,4'))
(Note("g'4"), Note('g,4'))
```

Iterate leaf pairs left-to-right and top-to-bottom.

Return generator.

`iterationtools.iterate_leaves_in_expr`

`iterationtools.iterate_leaves_in_expr` (*expr*, *reverse=False*, *start=0*, *stop=None*)

New in version 2.10. Iterate leaves forward in *expr*:

```
>>> staff = Staff("abj: | 2/8 c'8 d'8 || 2/8 e'8 f'8 || 2/8 g'8 a'8 |")
>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
}
```



```

        g'8
        a'8
    }
}

```

```

>>> for leaf in iterationtools.iterate_leaves_in_expr(staff):
...     leaf
...
Note("c'8")
Note("d'8")
Note("e'8")
Note("f'8")
Note("g'8")
Note("a'8")

```

Use the optional *start* and *stop* keyword parameters to control the start and stop indices of iteration.

```

>>> for leaf in iterationtools.iterate_leaves_in_expr(staff, start=3):
...     leaf
...
Note("f'8")
Note("g'8")
Note("a'8")

>>> for leaf in iterationtools.iterate_leaves_in_expr(staff, start=0, stop=3):
...     leaf
...
Note("c'8")
Note("d'8")
Note("e'8")

>>> for leaf in iterationtools.iterate_leaves_in_expr(staff, start=2, stop=4):
...     leaf
...
Note("e'8")
Note("f'8")

```

Iterate leaves backward in *expr*:

```

>>> for leaf in iterationtools.iterate_leaves_in_expr(staff, reverse=True):
...     leaf
...
Note("a'8")
Note("g'8")
Note("f'8")
Note("e'8")
Note("d'8")
Note("c'8")

>>> for leaf in iterationtools.iterate_leaves_in_expr(staff, start=3, reverse=True):
...     leaf
...
Note("e'8")
Note("d'8")
Note("c'8")

>>> for leaf in iterationtools.iterate_leaves_in_expr(staff, start=0, stop=3, reverse=True):
...     leaf
...

```

```
Note("a'8")
Note("g'8")
Note("f'8")
```

```
>>> for leaf in iterationtools.iterate_leaves_in_expr(staff, start=2, stop=4, reverse=True):
...     leaf
...
Note("f'8")
Note("e'8")
```

Ignore threads.

Return generator.

`iterationtools.iterate_measures_in_expr`

`iterationtools.iterate_measures_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.10. Iterate measures forward in *expr*:

```
>>> staff = Staff("abj: | 2/8 c'8 d'8 || 2/8 e'8 f'8 || 2/8 g'8 a'8 |")

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
}

>>> for measure in iterationtools.iterate_measures_in_expr(staff):
...     measure
...
Measure(2/8, [c'8, d'8])
Measure(2/8, [e'8, f'8])
Measure(2/8, [g'8, a'8])
```

Use the optional *start* and *stop* keyword parameters to control the start and stop indices of iteration.

```
>>> for measure in iterationtools.iterate_measures_in_expr(staff, start=1):
...     measure
...
Measure(2/8, [e'8, f'8])
Measure(2/8, [g'8, a'8])

>>> for measure in iterationtools.iterate_measures_in_expr(staff, start=0, stop=2):
...     measure
...
Measure(2/8, [c'8, d'8])
Measure(2/8, [e'8, f'8])
```

Iterate measures backward in *expr*:

```
>>> for measure in iterationtools.iterate_measures_in_expr(staff, reverse=True):
...     measure
...
Measure(2/8, [g'8, a'8])
Measure(2/8, [e'8, f'8])
Measure(2/8, [c'8, d'8])
```

Use the optional *start* and *stop* keyword parameters to control indices of iteration.

```
>>> for measure in iterationtools.iterate_measures_in_expr(staff, start=1, reverse=True):
...     measure
...
Measure(2/8, [e'8, f'8])
Measure(2/8, [c'8, d'8])

>>> for measure in iterationtools.iterate_measures_in_expr(staff, start=0, stop=2, reverse=True):
...     measure
...
Measure(2/8, [g'8, a'8])
Measure(2/8, [e'8, f'8])
```

Ignore threads.

Return generator.

iterationtools.iterate_namesakes_from_component

`iterationtools.iterate_namesakes_from_component` (*component*, *reverse=False*, *start=0*,
stop=None)

New in version 1.1. Iterate namesakes forward from *component*:

```
>>> container = Container(Staff(notetools.make_repeated_notes(2)) * 2)
>>> container.is_parallel = True
>>> container[0].name = 'staff 1'
>>> container[1].name = 'staff 2'
>>> score = Score([])
>>> score.is_parallel = False
>>> score.extend(container * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(score)

>>> f(score)
\new Score {
  <<
    \context Staff = "staff 1" {
      c'8
      d'8
    }
    \context Staff = "staff 2" {
      e'8
      f'8
    }
  >>
  <<
    \context Staff = "staff 1" {
      g'8
      a'8
    }
  >>
```

```

        }
        \context Staff = "staff 2" {
            b'8
            c''8
        }
    >>
}

>>> for staff in iterationtools.iterate_namesakes_from_component(score[0][0]):
...     print staff.lilypond_format
...
\context Staff = "staff 1" {
    c'8
    d'8
}
\context Staff = "staff 1" {
    g'8
    a'8
}

```

Iterate namesakes backward from *component*:

```

::

>>> for staff in iterationtools.iterate_namesakes_from_component(score[-1][0], reverse=True):
...     print staff.lilypond_format
...
\context Staff = "staff 1" {
    g'8
    a'8
}
\context Staff = "staff 1" {
    c'8
    d'8
}

```

Return generator.

iterationtools.iterate_notes_and_chords_in_expr

`iterationtools.iterate_notes_and_chords_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.10. Iterate notes and chords forward in *expr*:

```

>>> staff = Staff("<e' g' c''>8 a'8 r8 <d' f' b'>8 r2")

>>> f(staff)
\new Staff {
    <e' g' c''>8
    a'8
    r8
    <d' f' b'>8
    r2
}

>>> for leaf in iterationtools.iterate_notes_and_chords_in_expr(staff):
...     leaf
Chord("<e' g' c''>8")

```

```
Note("a'8")
Chord("<d' f' b'>8")
```

Iterate notes and chords backward in *expr*:

```
>>> for leaf in iterationtools.iterate_notes_and_chords_in_expr(staff, reverse=True):
...     leaf
Chord("<d' f' b'>8")
Note("a'8")
Chord("<e' g' c' '>8")
```

Ignore threads.

Return generator.

iterationtools.iterate_notes_in_expr

`iterationtools.iterate_notes_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.10. Yield left-to-right notes in *expr*:

```
>>> staff = Staff()
>>> staff.append(Measure((2, 8), "c'8 d'8"))
>>> staff.append(Measure((2, 8), "e'8 f'8"))
>>> staff.append(Measure((2, 8), "g'8 a'8"))

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
}

>>> for note in iterationtools.iterate_notes_in_expr(staff):
...     note
...
Note("c'8")
Note("d'8")
Note("e'8")
Note("f'8")
Note("g'8")
Note("a'8")
```

Use optional *start* and *stop* keyword parameters to control start and stop indices of iteration:

```
>>> for note in iterationtools.iterate_notes_in_expr(staff, start=3):
...     note
...
Note("f'8")
```

```
Note("g'8")
Note("a'8")

>>> for note in iterationtools.iterate_notes_in_expr(staff, start=0, stop=3):
...     note
...
Note("c'8")
Note("d'8")
Note("e'8")

>>> for note in iterationtools.iterate_notes_in_expr(staff, start=2, stop=4):
...     note
...
Note("e'8")
Note("f'8")
```

Yield right-to-left notes in *expr*:

```
>>> staff = Staff()
>>> staff.append(Measure((2, 8), "c'8 d'8"))
>>> staff.append(Measure((2, 8), "e'8 f'8"))
>>> staff.append(Measure((2, 8), "g'8 a'8"))

>>> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        e'8
        f'8
    }
    {
        g'8
        a'8
    }
}

>>> for note in iterationtools.iterate_notes_in_expr(staff, reverse=True):
...     note
...
Note("a'8")
Note("g'8")
Note("f'8")
Note("e'8")
Note("d'8")
Note("c'8")
```

Use optional *start* and *stop* keyword parameters to control indices of iteration:

```
>>> for note in iterationtools.iterate_notes_in_expr(staff, reverse=True, start=3):
...     note
...
Note("e'8")
Note("d'8")
Note("c'8")
```

```

>>> for note in iterationtools.iterate_notes_in_expr(staff, reverse=True, start=0, stop=3):
...     note
...
Note("a'8")
Note("g'8")
Note("f'8")

>>> for note in iterationtools.iterate_notes_in_expr(staff, reverse=True, start=2, stop=4):
...     note
...
Note("f'8")
Note("e'8")

```

Ignore threads.

Return generator.

iterationtools.iterate_rests_in_expr

`iterationtools.iterate_rests_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.10. Iterate rests forward in *expr*:

```

>>> staff = Staff("<e' g' c''>8 a'8 r8 <d' f' b'>8 r2")

>>> f(staff)
\new Staff {
    <e' g' c''>8
    a'8
    r8
    <d' f' b'>8
    r2
}

>>> for rest in iterationtools.iterate_rests_in_expr(staff):
...     rest
Rest('r8')
Rest('r2')

```

Iterate rests backward in *expr*:

```

>>> for rest in iterationtools.iterate_rests_in_expr(staff, reverse=True):
...     rest
Rest('r2')
Rest('r8')

```

Ignore threads.

Return generator.

iterationtools.iterate_scores_in_expr

`iterationtools.iterate_scores_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.10. Iterate scores forward in *expr*:

```

>>> score_1 = Score([Staff("c'8 d'8 e'8 f'8")])
>>> score_2 = Score([Staff("c'1"), Staff("g'1")])
>>> scores = [score_1, score_2]

```

```
>>> for score in iterationtools.iterate_scores_in_expr(scores):
...     score
Score<<1>>
Score<<2>>
```

Iterate scores backward in *expr*:

```
::
```

```
>>> for score in iterationtools.iterate_scores_in_expr(scores, reverse=True):
...     score
Score<<2>>
Score<<1>>
```

Ignore threads.

Return generator.

`iterationtools.iterate_semantic_voices_in_expr`

```
iterationtools.iterate_semantic_voices_in_expr(expr, reverse=False, start=0,
                                              stop=None)
```

New in version 2.0. Iterate semantic voices forward in *expr*:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(
...     [(3, 8), (5, 16), (5, 16)])
>>> meter_voice = Voice(measures)
>>> meter_voice.name = 'TimeSignatuerVoice'
>>> meter_voice.is_nonsemantic = True
>>> music_voice = Voice("c'4. d'4 e'16 f'4 g'16")
>>> music_voice.name = 'MusicVoice'
>>> staff = Staff([meter_voice, music_voice])
>>> staff.is_parallel = True

>>> f(staff)
\new Staff <<
  \context Voice = "TimeSignatuerVoice" {
    {
      \time 3/8
      s1 * 3/8
    }
    {
      \time 5/16
      s1 * 5/16
    }
    {
      s1 * 5/16
    }
  }
  \context Voice = "MusicVoice" {
    c'4.
    d'4
    e'16
    f'4
    g'16
  }
>>
```



```
>>> for voice in iterationtools.iterate_semantic_voices_in_expr(staff):
...     voice
Voice-"MusicVoice"{5}
```

Iterate semantic voices backward in *expr*:

```
>>> for voice in iterationtools.iterate_semantic_voices_in_expr(staff, reverse=True):
...     voice
Voice-"MusicVoice"{5}
```

Return generator.

iterationtools.iterate_skips_in_expr

`iterationtools.iterate_skips_in_expr(expr, reverse=False, start=0, stop=None)`
 New in version 2.10. Iterate skips forward in *expr*:

```
>>> staff = Staff("<e' g' c''>8 a'8 s8 <d' f' b'>8 s2")

>>> f(staff)
\new Staff {
    <e' g' c''>8
    a'8
    s8
    <d' f' b'>8
    s2
}
```

```
>>> for skip in iterationtools.iterate_skips_in_expr(staff):
...     skip
Skip('s8')
Skip('s2')
```

Iterate skips backwards in *expr*:

```
>>> for skip in iterationtools.iterate_skips_in_expr(staff, reverse=True):
...     skip
Skip('s2')
Skip('s8')
```

Ignore threads.

Return generator.

iterationtools.iterate_staves_in_expr

`iterationtools.iterate_staves_in_expr(expr, reverse=False, start=0, stop=None)`
 New in version 2.10. Iterate staves forward in *expr*:

```
>>> score = Score(4 * Staff([]))

>>> f(score)
\new Score <<
  \new Staff {
  }
  \new Staff {
  }
}
```

```

        \new Staff {
        }
        \new Staff {
        }
>>

>>> for staff in iterationtools.iterate_staves_in_expr(score):
...     staff
...
Staff{}
Staff{}
Staff{}
Staff{}

```

Iterate staves backward in *expr*:

```

::

>>> for staff in iterationtools.iterate_staves_in_expr(score, reverse=True):
...     staff
...
Staff{}
Staff{}
Staff{}
Staff{}

```

Return generator.

iterationtools.iterate_thread_from_component

`iterationtools.iterate_thread_from_component` (*component*, *klass=None*, *reverse=False*)

New in version 2.10. Iterate thread forward from *component* and yield instances of *klass*:

```

>>> container = Container(Voice(notetools.make_repeated_notes(2)) * 2)
>>> container.is_parallel = True
>>> container[0].name = 'voice 1'
>>> container[1].name = 'voice 2'
>>> staff = Staff(container * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> f(staff)
\new Staff {
  <<
    \context Voice = "voice 1" {
      c'8
      d'8
    }
    \context Voice = "voice 2" {
      e'8
      f'8
    }
  >>
  <<
    \context Voice = "voice 1" {
      g'8
      a'8
    }
    \context Voice = "voice 2" {

```

```

        b'8
        c''8
    }
>>
}

```

Starting from the first leaf in score.

```

>>> for x in iterationtools.iterate_thread_from_component(staff.leaves[0], Note):
...     x
...
Note("c'8")
Note("d'8")
Note("g'8")
Note("a'8")

```

Starting from the second leaf in score.

```

>>> for x in iterationtools.iterate_thread_from_component(staff.leaves[1], Note):
...     x
...
Note("d'8")
Note("g'8")
Note("a'8")

```

Yield all components in thread.

```

>>> for x in iterationtools.iterate_thread_from_component(staff.leaves[0]):
...     x
...
Note("c'8")
Voice-"voice 1"{2}
Note("d'8")
Voice-"voice 1"{2}
Note("g'8")
Note("a'8")

```

Iterate thread backward from *component* and yield instances of *klass*, starting from the last leaf in score.

```

>>> for x in iterationtools.iterate_thread_from_component(staff.leaves[-1], Note, reverse=True):
...     x
...
Note("c'8")
Note("b'8")
Note("f'8")
Note("e'8")

```

Yield all components in thread:

```

>>> for x in iterationtools.iterate_thread_from_component(staff.leaves[-1], reverse=True):
...     x
...
Note("c'8")
Voice-"voice 2"{2}
Note("b'8")
Voice-"voice 2"{2}
Note("f'8")
Note("e'8")

```

Return generator.

iterationtools.iterate_thread_in_expr

`iterationtools.iterate_thread_in_expr(expr, klass, containment_signature, reverse=False)`

New in version 2.10. Yield left-to-right instances of *klass* in *expr* with *containment_signature*:

```
>>> container = Container(Voice(notetools.make_repeated_notes(2)) * 2)
>>> container.is_parallel = True
>>> container[0].name = 'voice 1'
>>> container[1].name = 'voice 2'
>>> staff = Staff(container * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> f(staff)
\new Staff {
  <<
    \context Voice = "voice 1" {
      c'8
      d'8
    }
    \context Voice = "voice 2" {
      e'8
      f'8
    }
  >>
  <<
    \context Voice = "voice 1" {
      g'8
      a'8
    }
    \context Voice = "voice 2" {
      b'8
      c''8
    }
  >>
}

>>> signature = componenttools.component_to_containment_signature(staff.leaves[0])
>>> for x in iterationtools.iterate_thread_in_expr(staff, Note, signature):
...     x
...
Note("c'8")
Note("d'8")
Note("g'8")
Note("a'8")
```

Return generator.

iterationtools.iterate_timeline_from_component

`iterationtools.iterate_timeline_from_component(expr, klass=None, reverse=False)`

New in version 2.10. Iterate timeline forward from *component*:

```
>>> score = Score([])
>>> score.append(Staff(notetools.make_repeated_notes(4, Duration(1, 4))))
>>> score.append(Staff(notetools.make_repeated_notes(4)))
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(score)
```

```
>>> f(score)
\new Score <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
  \new Staff {
    g'8
    a'8
    b'8
    c''8
  }
>>

>>> for leaf in iterationtools.iterate_timeline_from_component(score[1][2]):
...     leaf
...
Note("b'8")
Note("c''8")
Note("e'4")
Note("f'4")
```

Iterate timeline backward from *component*:

```
::

>>> for leaf in iterationtools.iterate_timeline_from_component(score[1][2], reverse=True):
...     leaf
...
Note("b'8")
Note("c'4")
Note("a'8")
Note("g'8")
```

Yield components sorted backward by score offset stop time when *reverse* is True.

Iterate leaves when *klass* is none.

Todo

optimize to avoid behind-the-scenes full-score traversal.

`iterationtools.iterate_timeline_in_expr`

`iterationtools.iterate_timeline_in_expr` (*expr*, *klass=None*, *reverse=False*)

New in version 2.10. Iterate timeline forward in *expr*:

```
>>> score = Score([])
>>> score.append(Staff(notetools.make_repeated_notes(4, Duration(1, 4))))
>>> score.append(Staff(notetools.make_repeated_notes(4)))
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(score)

>>> f(score)
\new Score <<
  \new Staff {
```

```

        c'4
        d'4
        e'4
        f'4
    }
    \new Staff {
        g'8
        a'8
        b'8
        c''8
    }
>>

>>> for leaf in iterationtools.iterate_timeline_in_expr(score):
...     leaf
...
Note("c'4")
Note("g'8")
Note("a'8")
Note("d'4")
Note("b'8")
Note("c''8")
Note("e'4")
Note("f'4")

```

Iterate timeline backward in *expr*:

```

::

>>> for leaf in iterationtools.iterate_timeline_in_expr(score, reverse=True):
...     leaf
...
Note("f'4")
Note("e'4")
Note("d'4")
Note("c''8")
Note("b'8")
Note("c'4")
Note("a'8")
Note("g'8")

```

Iterate leaves when *klass* is none.

Todo

optimize to avoid behind-the-scenes full-score traversal.

iterationtools.iterate_tuplets_in_expr

`iterationtools.iterate_tuplets_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.10. Iterate tuplets forward in *expr*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
>>> Tuplet(Fraction(2, 3), staff[:3])
Tuplet(2/3, [c'8, d'8, e'8])
>>> Tuplet(Fraction(2, 3), staff[-3:])
Tuplet(2/3, [a'8, b'8, c''8])

```

```

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8
    d'8
    e'8
  }
  f'8
  g'8
  \times 2/3 {
    a'8
    b'8
    c''8
  }
}

>>> for tuplet in iterationtools.iterate_tuplets_in_expr(staff):
...     tuplet
...
Tuplet(2/3, [c'8, d'8, e'8])
Tuplet(2/3, [a'8, b'8, c''8])

```

Iterate tuplets backward in *expr*:

```

>>> for tuplet in iterationtools.iterate_tuplets_in_expr(staff, reverse=True):
...     tuplet
...
Tuplet(2/3, [a'8, b'8, c''8])
Tuplet(2/3, [c'8, d'8, e'8])

```

Ignore threads.

Return generator.

iterationtools.iterate_voices_in_expr

`iterationtools.iterate_voices_in_expr(expr, reverse=False, start=0, stop=None)`

New in version 2.0. Iterate voices forward in *expr*:

```

>>> voice_1 = Voice("c'8 d'8 e'8 f'8")
>>> voice_2 = Voice("c'4 b4")
>>> staff = Staff([voice_1, voice_2])
>>> staff.is_parallel = True

>>> f(staff)
\new Staff <<
  \new Voice {
    c'8
    d'8
    e'8
    f'8
  }
  \new Voice {
    c'4
    b4
  }
>>

```

```
>>> for voice in iterationtools.iterate_voices_in_expr(staff):
...     voice
Voice{4}
Voice{2}
```

Iterate voices backward in *expr*:

```
::
```

```
>>> for voice in iterationtools.iterate_voices_in_expr(staff, reverse=True):
...     voice
Voice{2}
Voice{4}
```

Return generator.

labeltools

functions

labeltools.color_chord_note_heads_in_expr_by_pitch_class_color_map

`labeltools.color_chord_note_heads_in_expr_by_pitch_class_color_map`(*chord*,
color_map)

New in version 2.0. Color *chord* note heads by pitch-class *color_map*:

```
>>> chord = Chord([12, 14, 18, 21, 23], (1, 4))

>>> pitches = [[-12, -10, 4], [-2, 8, 11, 17], [19, 27, 30, 33, 37]]
>>> colors = ['red', 'blue', 'green']
>>> color_map = pitchtools.NumberedChromaticPitchClassColorMap(pitches, colors)

>>> labeltools.color_chord_note_heads_in_expr_by_pitch_class_color_map(chord, color_map)
Chord("<c'' d'' fs'' a'' b''>4")

>>> f(chord)
<
  \tweak #'color #red
  c''
  \tweak #'color #red
  d''
  \tweak #'color #green
  fs''
  \tweak #'color #green
  a''
  \tweak #'color #blue
  b''
>4
```

Also works on notes:

```
>>> note = Note("c'4")

>>> labeltools.color_chord_note_heads_in_expr_by_pitch_class_color_map(note, color_map)
Note("c'4")
```



```
>>> f(note)
\once \override NoteHead #'color = #red
c'4
```

When *chord* is neither a chord nor note return *chord* unchanged:

```
>>> staff = Staff([])

>>> labeltools.color_chord_note_heads_in_expr_by_pitch_class_color_map(staff, color_map)
Staff{}
```

Return *chord*. Changed in version 2.0: renamed `chordtools.color_note_heads_by_pc()` to `labeltools.color_chord_note_heads_in_expr_by_pitch_class_color_map()`.

labeltools.color_contents_of_container

`labeltools.color_contents_of_container(container, color)`

New in version 2.0. Color contents of *container*:

```
>>> measure = Measure((2, 8), "c'8 d'8")

>>> labeltools.color_contents_of_container(measure, 'red')
Measure(2/8, [c'8, d'8])

>>> f(measure)
{
  \override Accidental #'color = #red
  \override Beam #'color = #red
  \override Dots #'color = #red
  \override NoteHead #'color = #red
  \override Rest #'color = #red
  \override Stem #'color = #red
  \override TupletBracket #'color = #red
  \override TupletNumber #'color = #red
  \time 2/8
  c'8
  d'8
  \revert Accidental #'color
  \revert Beam #'color
  \revert Dots #'color
  \revert NoteHead #'color
  \revert Rest #'color
  \revert Stem #'color
  \revert TupletBracket #'color
  \revert TupletNumber #'color
}
```

Return `none`. Changed in version 2.0: renamed `containertools.contents_color()` to `labeltools.color_contents_of_container()`.

labeltools.color_leaf

`labeltools.color_leaf(leaf, color)`

New in version 2.0. Color note:

```
>>> note = Note("c'4")

>>> labeltools.color_leaf(note, 'red')
Note("c'4")

>>> f(note)
\once \override Accidental #'color = #red
\once \override Dots #'color = #red
\once \override NoteHead #'color = #red
c'4
```

Color rest:

```
>>> rest = Rest('r4')

>>> labeltools.color_leaf(rest, 'red')
Rest('r4')

>>> f(rest)
\once \override Dots #'color = #red
\once \override Rest #'color = #red
r4
```

Color chord:

```
>>> chord = Chord("<c' e' bf'>4")

>>> labeltools.color_leaf(chord, 'red')
Chord("<c' e' bf'>4")

>>> f(chord)
\once \override Accidental #'color = #red
\once \override Dots #'color = #red
\once \override NoteHead #'color = #red
<c' e' bf'>4
```

Return *leaf*.

labeltools.color_leaves_in_expr

labeltools.**color_leaves_in_expr**(*expr*, *color*)

New in version 2.0. Color leaves in *expr*:

```
>>> staff = Staff("cs'8. [ r8. s8. <c' cs' a'>8. ]")

>>> f(staff)
\new Staff {
  cs'8. [
    r8.
    s8.
    <c' cs' a'>8. ]
}

>>> labeltools.color_leaves_in_expr(staff, 'red')

>>> f(staff)
\new Staff {
  \once \override Accidental #'color = #red
```

```

\once \override Dots #'color = #red
\once \override NoteHead #'color = #red
cs'8. [
\once \override Dots #'color = #red
\once \override Rest #'color = #red
r8.
s8.
\once \override Accidental #'color = #red
\once \override Dots #'color = #red
\once \override NoteHead #'color = #red
<c' cs' a'>8. ]
}

```

Return none.

labeltools.color_measure

`labeltools.color_measure(measure, color='red')`

New in version 2.0. Color *measure* with *color*:

```

>>> measure = Measure((2, 8), "c'8 d'8")

>>> f(measure)
{
  \time 2/8
  c'8
  d'8
}

>>> labeltools.color_measure(measure, 'red')
Measure(2/8, [c'8, d'8])

>>> f(measure)
{
  \override Beam #'color = #red
  \override Dots #'color = #red
  \override NoteHead #'color = #red
  \override Staff.TimeSignature #'color = #red
  \override Stem #'color = #red
  \time 2/8
  c'8
  d'8
  \revert Beam #'color
  \revert Dots #'color
  \revert NoteHead #'color
  \revert Staff.TimeSignature #'color
  \revert Stem #'color
}

```

Return colored *measure*.

Color names appear in LilyPond Learning Manual appendix B.5.

labeltools.color_nonbinary_measures_in_expr

`labeltools.color_nonbinary_measures_in_expr(expr, color='red')`

New in version 2.0. Color nonbinary measures in *expr* with *color*:

```
>>> staff = Staff(Measure((2, 8), "c'8 d'8") * 2)
>>> measuretools.scale_measure_denominator_and_adjust_measure_contents(staff[1], 3)
Measure(3/12, [c'8., d'8.])

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 3/12
    \scaleDurations #'(2 . 3) {
      c'8.
      d'8.
    }
  }
}

>>> labeltools.color_nonbinary_measures_in_expr(staff, 'red')
[Measure(3/12, [c'8., d'8.])]

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \override Beam #'color = #red
    \override Dots #'color = #red
    \override NoteHead #'color = #red
    \override Staff.TimeSignature #'color = #red
    \override Stem #'color = #red
    \time 3/12
    \scaleDurations #'(2 . 3) {
      c'8.
      d'8.
    }
    \revert Beam #'color
    \revert Dots #'color
    \revert NoteHead #'color
    \revert Staff.TimeSignature #'color
    \revert Stem #'color
  }
}
```

Return list of measures colored.

Color names appear in LilyPond Learning Manual appendix B.5.

labeltools.color_note_head_by_numbered_chromatic_pitch_class_color_map

`labeltools.color_note_head_by_numbered_chromatic_pitch_class_color_map(pitch_carrier)`
 Color *pitch_carrier* note head:

```
>>> note = Note("c'4")

>>> labeltools.color_note_head_by_numbered_chromatic_pitch_class_color_map(note)
Note("c'4")

>>> f(note)
\once \override NoteHead #'color = #(x11-color 'red)
c'4
```

Numbered chromatic pitch-class color map:

```
0: red
1: MediumBlue
2: orange
3: LightSlateBlue
4: ForestGreen
5: MediumOrchid
6: firebrick
7: DeepPink
8: DarkOrange
9: IndianRed
10: CadetBlue
11: SeaGreen
12: LimeGreen
```

Numbered chromatic pitch-class color map can not be changed.

Raise type error when *pitch_carrier* is not a pitch carrier.

Raise extra pitch error when *pitch_carrier* carries more than 1 note head.

Raise missing pitch error when *pitch_carrier* carries no note head.

Return *pitch_carrier*. Changed in version 2.0: renamed `pitchtools.color_by_pc()` to `labeltools.color_note_head_by_numbered_chromatic_pitch_class_color_map()`. Changed in version 2.0: renamed `notetools.color_note_head_by_numeric_chromatic_pitch_class_color_map()` to `labeltools.color_note_head_by_numbered_chromatic_pitch_class_color_map()`.

labeltools.label_leaves_in_expr_with_inversion_equivalent_chromatic_interval_classes

`labeltools.label_leaves_in_expr_with_inversion_equivalent_chromatic_interval_classes` (*expr*, *markup*)

New in version 2.0. Label leaves in *expr* with inversion-equivalent chromatic interval classes:

```
>>> notes = notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
>>> staff = Staff(notes)
>>> labeltools.label_leaves_in_expr_with_inversion_equivalent_chromatic_interval_classes(
...     staff)

>>> f(staff)
\new Staff {
  c'8 ^ \markup { 1 }
  cs''8 ^ \markup { 2 }
  b'8 ^ \markup { 2 }
  af8 ^ \markup { 2 }
  bf,8 ^ \markup { 1 }
  b,8 ^ \markup { 2 }
  a'8 ^ \markup { 1 }
  bf'8 ^ \markup { 4 }
```

```

    fs'8 ^ \markup { 1 }
    f'8
}

```

Return none.

labeltools.label_leaves_in_expr_with_leaf_depth

`labeltools.label_leaves_in_expr_with_leaf_depth(expr, markup_direction=Down)`

New in version 1.1. Label leaves in *expr* with leaf depth:

```

>>> staff = Staff("c'8 d'8 e'8 f'8 g'8")
>>> tuplettools.FixedDurationTuplet(Duration(2, 8), staff[-3:])
FixedDurationTuplet(1/4, [e'8, f'8, g'8])
>>> labeltools.label_leaves_in_expr_with_leaf_depth(staff)
>>> f(staff)
\new Staff {
  c'8 _ \markup { \small 1 }
  d'8 _ \markup { \small 1 }
  \times 2/3 {
    e'8 _ \markup { \small 2 }
    f'8 _ \markup { \small 2 }
    g'8 _ \markup { \small 2 }
  }
}

```

Changed in version 2.0: renamed `label.leaf_depth()` to `labeltools.label_leaves_in_expr_with_leaf_depth()`.
Return none.

labeltools.label_leaves_in_expr_with_leaf_durations

`labeltools.label_leaves_in_expr_with_leaf_durations(expr, markup_direction=Down)`

New in version 1.1. Label leaves in *expr* with leaf durations:

```

>>> tuplet = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8 e'8")
>>> labeltools.label_leaves_in_expr_with_leaf_durations(tuplet)
>>> f(tuplet)
\times 2/3 {
  c'8 _ \markup { \column { \small 1/8 \small 1/12 } }
  d'8 _ \markup { \column { \small 1/8 \small 1/12 } }
  e'8 _ \markup { \column { \small 1/8 \small 1/12 } }
}

```

Label both written duration and prolated duration.

Return none.

labeltools.label_leaves_in_expr_with_leaf_indices

`labeltools.label_leaves_in_expr_with_leaf_indices(expr, markup_direction=Down)`

New in version 2.0. Label leaves in *expr* with leaf indices:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> labeltools.label_leaves_in_expr_with_leaf_indices(staff)
>>> f(staff)

```

```

\new Staff {
  c'8 _ \markup { \small 0 }
  d'8 _ \markup { \small 1 }
  e'8 _ \markup { \small 2 }
  f'8 _ \markup { \small 3 }
}

```

Return none.

labeltools.label_leaves_in_expr_with_leaf_numbers

labeltools.label_leaves_in_expr_with_leaf_numbers (*expr*, *markup_direction=Down*)

New in version 1.1. Label leaves in *expr* with leaf numbers:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> labeltools.label_leaves_in_expr_with_leaf_numbers(staff)
>>> f(staff)
\new Staff {
  c'8 _ \markup { \small 1 }
  d'8 _ \markup { \small 2 }
  e'8 _ \markup { \small 3 }
  f'8 _ \markup { \small 4 }
}

```

Number leaves starting from 1. Changed in version 2.0: renamed `label.leaf_numbers()` to `labeltools.label_leaves_in_expr_with_leaf_numbers()`. Return none.

labeltools.label_leaves_in_expr_with_melodic_chromatic_interval_classes

labeltools.label_leaves_in_expr_with_melodic_chromatic_interval_classes (*expr*,
markup_direction=Up)

New in version 2.0. Label leaves in *expr* with melodic chromatic interval classes:

```

>>> notes = notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
>>> staff = Staff(notes)
>>> labeltools.label_leaves_in_expr_with_melodic_chromatic_interval_classes(staff)

>>> f(staff)
\new Staff {
  c'8 ^ \markup { +1 }
  cs'''8 ^ \markup { -2 }
  b'8 ^ \markup { -2 }
  af8 ^ \markup { -10 }
  bf,8 ^ \markup { +1 }
  b,8 ^ \markup { +10 }
  a'8 ^ \markup { +1 }
  bf'8 ^ \markup { -4 }
  fs'8 ^ \markup { -1 }
  f'8
}

```

Return none.

labeltools.label_leaves_in_expr_with_melodic_chromatic_intervals

`labeltools.label_leaves_in_expr_with_melodic_chromatic_intervals`(*expr*,
markup_direction=Up)

New in version 2.0. Label leaves in *expr* with melodic chromatic intervals:

```
>>> notes = notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
>>> staff = Staff(notes)
>>> labeltools.label_leaves_in_expr_with_melodic_chromatic_intervals(staff)

>>> f(staff)
\new Staff {
  c'8 ^ \markup { +25 }
  cs'''8 ^ \markup { -14 }
  b'8 ^ \markup { -15 }
  af8 ^ \markup { -10 }
  bf,8 ^ \markup { +1 }
  b,8 ^ \markup { +22 }
  a'8 ^ \markup { +1 }
  bf'8 ^ \markup { -4 }
  fs'8 ^ \markup { -1 }
  f'8
}
```

Return none.

labeltools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes

`labeltools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes`(*expr*,
markup_direction=Up)

New in version 2.0. Label leaves in *expr* with melodic counterpoint interval classes:

```
>>> notes = notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
>>> staff = Staff(notes)
>>> labeltools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes(staff)

>>> f(staff)
\new Staff {
  c'8 ^ \markup { +8 }
  cs'''8 ^ \markup { -2 }
  b'8 ^ \markup { -2 }
  af8 ^ \markup { -7 }
  bf,8 ^ \markup { +1 }
  b,8 ^ \markup { +7 }
  a'8 ^ \markup { +2 }
  bf'8 ^ \markup { -4 }
  fs'8 ^ \markup { +1 }
  f'8
}
```

Return none.

labeltools.label_leaves_in_expr_with_melodic_counterpoint_intervals

`labeltools.label_leaves_in_expr_with_melodic_counterpoint_intervals`(*expr*,
markup_direction=Up)

New in version 2.0. Label leaves in *expr* with melodic counterpoint intervals:

```
>>> notes = notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
>>> staff = Staff(notes)
>>> labeltools.label_leaves_in_expr_with_melodic_counterpoint_intervals(staff)

>>> f(staff)
\new Staff {
  c'8 ^ \markup { +15 }
  cs'''8 ^ \markup { -9 }
  b'8 ^ \markup { -9 }
  af8 ^ \markup { -7 }
  bf,8 ^ \markup { 1 }
  b,8 ^ \markup { +14 }
  a'8 ^ \markup { +2 }
  bf'8 ^ \markup { -4 }
  fs'8 ^ \markup { 1 }
  f'8
}
```

Return none.

labeltools.label_leaves_in_expr_with_melodic_diatonic_interval_classes

`labeltools.label_leaves_in_expr_with_melodic_diatonic_interval_classes`(*expr*,
markup_direction=Up)

New in version 2.0. Label leaves in *expr* with melodic diatonic interval classes:

```
>>> notes = notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
>>> staff = Staff(notes)
>>> labeltools.label_leaves_in_expr_with_melodic_diatonic_interval_classes(staff)

>>> f(staff)
\new Staff {
  c'8 ^ \markup { +aug8 }
  cs'''8 ^ \markup { -M2 }
  b'8 ^ \markup { -aug2 }
  af8 ^ \markup { -m7 }
  bf,8 ^ \markup { aug1 }
  b,8 ^ \markup { +m7 }
  a'8 ^ \markup { +m2 }
  bf'8 ^ \markup { -dim4 }
  fs'8 ^ \markup { aug1 }
  f'8
}
```

Return none.

labeltools.label_leaves_in_expr_with_melodic_diatonic_intervals

`labeltools.label_leaves_in_expr_with_melodic_diatonic_intervals`(*expr*,
markup_direction=Up)

New in version 2.0. Label leaves in *expr* with melodic diatonic intervals:

```
>>> notes = notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
>>> staff = Staff(notes)
>>> labeltools.label_leaves_in_expr_with_melodic_diatonic_intervals(staff)

>>> f(staff)
\new Staff {
    c'8 ^ \markup { +aug15 }
    cs''8 ^ \markup { -M9 }
    b'8 ^ \markup { -aug9 }
    af8 ^ \markup { -m7 }
    bf,8 ^ \markup { +aug1 }
    b,8 ^ \markup { +m14 }
    a'8 ^ \markup { +m2 }
    bf'8 ^ \markup { -dim4 }
    fs'8 ^ \markup { -aug1 }
    f'8
}
```

Return none.

labeltools.label_leaves_in_expr_with_pitch_class_numbers

`labeltools.label_leaves_in_expr_with_pitch_class_numbers`(*expr*,
number=True,
color=False,
markup_direction=Down)

New in version 1.1. Label leaves in *expr* with pitch-class numbers:

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> labeltools.label_leaves_in_expr_with_pitch_class_numbers(t)
>>> print t.lilypond_format
\new Staff {
    c'8 _ \markup { \small 0 }
    d'8 _ \markup { \small 2 }
    e'8 _ \markup { \small 4 }
    f'8 _ \markup { \small 5 }
}
```

When *color=True* call `color_note_head_by_numbered_chromatic_pitch_class_color_map()`:

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> labeltools.label_leaves_in_expr_with_pitch_class_numbers(t, color=True, number=False)
>>> print t.lilypond_format
\new Staff {
    \once \override NoteHead #'color = #(x11-color 'red)
    c'8
    \once \override NoteHead #'color = #(x11-color 'orange)
    d'8
    \once \override NoteHead #'color = #(x11-color 'ForestGreen)
    e'8
    \once \override NoteHead #'color = #(x11-color 'MediumOrchid)
    f'8
}
```

You can set *number* and *color* at the same time. Changed in version 2.0: renamed `label.leaf_pcs()` to `labeltools.label_leaves_in_expr_with_pitch_class_numbers()`. Return `none`.

`labeltools.label_leaves_in_expr_with_pitch_numbers`

`labeltools.label_leaves_in_expr_with_pitch_numbers(expr, markup_direction=Down)`

New in version 1.1. Label leaves in *expr* with pitch numbers:

```
>>> staff = Staff(leaftools.make_leaves([None, 12, [13, 14, 15], None], [(1, 4)]))
>>> labeltools.label_leaves_in_expr_with_pitch_numbers(staff)
>>> f(staff)
\new Staff {
  r4
  c'4 _ \markup { \small 12 }
  <cs'' d'' ef''>4 _ \markup { \column { \small 15 \small 14 \small 13 } }
  r4
}
```

Return `none`. Changed in version 2.0: renamed `label.leaf_pitch_numbers()` to `labeltools.label_leaves_in_expr_with_pitch_numbers()`.

`labeltools.label_leaves_in_expr_with_prolated_leaf_duration`

`labeltools.label_leaves_in_expr_with_prolated_leaf_duration(expr, markup_direction=Down)`

New in version 1.1. Label leaves in *expr* with prolated leaf duration:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8 e'8")
>>> labeltools.label_leaves_in_expr_with_prolated_leaf_duration(tuplet)
>>> f(tuplet)
\times 2/3 {
  c'8 _ \markup { \small 1/12 }
  d'8 _ \markup { \small 1/12 }
  e'8 _ \markup { \small 1/12 }
}
```

Return `none`.

`labeltools.label_leaves_in_expr_with_tuplet_depth`

`labeltools.label_leaves_in_expr_with_tuplet_depth(expr, markup_direction=Down)`

New in version 1.1. Label leaves in *expr* with tuplet depth:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8")
>>> tuplettools.FixedDurationTuplet(Duration(2, 8), staff[-3:])
FixedDurationTuplet(1/4, [e'8, f'8, g'8])
>>> labeltools.label_leaves_in_expr_with_tuplet_depth(staff)
>>> f(staff)
\new Staff {
  c'8 _ \markup { \small 0 }
  d'8 _ \markup { \small 0 }
  \times 2/3 {
    e'8 _ \markup { \small 1 }
    f'8 _ \markup { \small 1 }
    g'8 _ \markup { \small 1 }
  }
}
```

```
}
}
```

Return `none`. Changed in version 2.0: renamed `label.leaf_depth_tuplet()` to `labeltools.label_leaves_in_expr_with_tuplet_depth()`.

labeltools.label_leaves_in_expr_with_written_leaf_duration

`labeltools.label_leaves_in_expr_with_written_leaf_duration`(*expr*,
markup_direction=Down)

New in version 1.1. Label leaves in *expr* with written leaf duration:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8 e'8")
>>> labeltools.label_leaves_in_expr_with_leaf_durations(tuplet)
>>> f(tuplet)
\times 2/3 {
    c'8 _ \markup { \column { \small 1/8 \small 1/12 } }
    d'8 _ \markup { \column { \small 1/8 \small 1/12 } }
    e'8 _ \markup { \column { \small 1/8 \small 1/12 } }
}
```

Return `none`.

labeltools.label_notes_in_expr_with_note_indices

`labeltools.label_notes_in_expr_with_note_indices`(*expr*, *markup_direction=Down*)

New in version 2.0. Label notes in *expr* with note indices:

```
>>> staff = Staff("c'8 d'8 r8 r8 g'8 a'8 r8 c''8")
>>> labeltools.label_notes_in_expr_with_note_indices(staff)

>>> f(staff)
\new Staff {
    c'8 _ \markup { \small 0 }
    d'8 _ \markup { \small 1 }
    r8
    r8
    g'8 _ \markup { \small 2 }
    a'8 _ \markup { \small 3 }
    r8
    c''8 _ \markup { \small 4 }
}
```

Return `none`.

labeltools.label_tie_chains_in_expr_with_prolated_tie_chain_duration

`labeltools.label_tie_chains_in_expr_with_prolated_tie_chain_duration`(*expr*,
markup_direction=Down)

Label tie chains in *expr* with prolated tie chain duration:

```
>>> staff = Staff(r"\times 2/3 { c'8 ~ c'8 c'8 ~ } c'8")
>>> labeltools.label_tie_chains_in_expr_with_prolated_tie_chain_duration(staff)
```

```
>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8 ~
    - \markup {
      \small
      1/6
    }
    c'8
    c'8 ~
    - \markup {
      \small
      5/24
    }
  }
  c'8
}
```

Return none.

labeltools.label_tie_chains_in_expr_with_tie_chain_durations

labeltools.label_tie_chains_in_expr_with_tie_chain_durations(*expr*,
markup_direction=Down)

Label tie chains in *expr* with both written tie chain duration and prolated tie chain duration:

```
>>> staff = Staff(r"\times 2/3 { c'8 ~ c'8 c'8 ~ } c'8")
>>> labeltools.label_tie_chains_in_expr_with_tie_chain_durations(staff)
```

```
>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8 ~
    - \markup {
      \column
      {
        \small
        1/4
        \small
        1/6
      }
    }
    c'8
    c'8 ~
    - \markup {
      \column
      {
        \small
        1/4
        \small
        5/24
      }
    }
  }
  c'8
}
```

Return none.

labeltools.label_tie_chains_in_expr_with_written_tie_chain_duration

`labeltools.label_tie_chains_in_expr_with_written_tie_chain_duration`(*expr*,
markup_direction=Down)

Label tie chains in *expr* with written tie chain duration.:

```
>>> staff = Staff(r"\times 2/3 { c'8 ~ c'8 c'8 ~ } c'8")
>>> labeltools.label_tie_chains_in_expr_with_written_tie_chain_duration(staff)

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8 ~
    _ \markup {
      \small
      1/4
    }
    c'8
    c'8 ~
    _ \markup {
      \small
      1/4
    }
  }
  c'8
}
```

Return none.

labeltools.label_vertical_moments_in_expr_with_chromatic_interval_classes

`labeltools.label_vertical_moments_in_expr_with_chromatic_interval_classes`(*expr*,
markup_direction=Down)

New in version 2.0. Label harmonic chromatic interval-classes of every vertical moment in *expr*:

```
>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "alto" g4 f4""")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "bass" c,2""")
>>> score.append(staff)

>>> labeltools.label_vertical_moments_in_expr_with_chromatic_interval_classes(
...     score)

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    _ \markup {
      \small
      \column
      {
```

```

                2
                7
            }
        }
    e'8
    f'8
    _ \markup {
        \small
        \column
        {
            5
            5
        }
    }
}
\new Staff {
    \clef "alto"
    g4
    f4
    _ \markup {
        \small
        \column
        {
            4
            5
        }
    }
}
\new Staff {
    \clef "bass"
    c,2
    _ \markup {
        \small
        \column
        {
            12
            7
        }
    }
}
>>

```

Return none.

labeltools.label_vertical_moments_in_expr_with_chromatic_intervals

`labeltools.label_vertical_moments_in_expr_with_chromatic_intervals` (*expr*, *markup_direction=Down*)

New in version 2.0. Label harmonic chromatic intervals of every vertical moment in *expr*:

```

>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "alto" g4 f4""")
>>> score.append(staff)

```

```
>>> staff = Staff(r"""\clef "bass" c,2""")
>>> score.append(staff)

>>> labeltools.label_vertical_moments_in_expr_with_chromatic_intervals(
...     score)

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    -- \markup {
      \small
      \column
      {
        26
        19
      }
    }
    e'8
    f'8
    -- \markup {
      \small
      \column
      {
        29
        17
      }
    }
  }
  \new Staff {
    \clef "alto"
    g4
    f4
    -- \markup {
      \small
      \column
      {
        28
        17
      }
    }
  }
  \new Staff {
    \clef "bass"
    c,2
    -- \markup {
      \small
      \column
      {
        24
        19
      }
    }
  }
}
>>
```

Return none.

labeltools.label_vertical_moments_in_expr_with_counterpoint_intervals

`labeltools.label_vertical_moments_in_expr_with_counterpoint_intervals`(*expr*,
markup_direction=Down)

New in version 2.0. Label counterpoint interval of every vertical moment in *expr*:

```
>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "alto" g4 f4""")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "bass" c,2""")
>>> score.append(staff)

>>> labeltools.label_vertical_moments_in_expr_with_counterpoint_intervals(score)

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    _ \markup {
      \small
      \column
      {
        2
        5
      }
    }
    e'8
    f'8
    _ \markup {
      \small
      \column
      {
        4
        4
      }
    }
  }
  \new Staff {
    \clef "alto"
    g4
    f4
    _ \markup {
      \small
      \column
      {
        3
        4
      }
    }
  }
  \new Staff {
    \clef "bass"
    c,2
```

```

        _ \markup {
            \small
            \column
            {
                8
                5
            }
        }
    }
>>

```

Return none.

`labeltools.label_vertical_moments_in_expr_with_diatonic_intervals`

`labeltools.label_vertical_moments_in_expr_with_diatonic_intervals`(*expr*,
markup_direction=Down)

New in version 2.0. Label diatonic intervals of every vertical moment in *expr*:

```

>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "alto" g4 f4""")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "bass" c,2""")
>>> score.append(staff)

>>> labeltools.label_vertical_moments_in_expr_with_diatonic_intervals(score)

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    _ \markup {
      \small
      \column
      {
        16
        12
      }
    }
    e'8
    f'8
    _ \markup {
      \small
      \column
      {
        18
        11
      }
    }
  }
  \new Staff {
    \clef "alto"

```

```

g4
f4
    _ \markup {
      \small
      \column
      {
        17
        11
      }
    }
  }
\new Staff {
  \clef "bass"
  c,2
  _ \markup {
    \small
    \column
    {
      15
      12
    }
  }
}
>>

```

Return none.

labeltools.label_vertical_moments_in_expr_with_interval_class_vectors

`labeltools.label_vertical_moments_in_expr_with_interval_class_vectors` (*expr*,
markup_direction=Down)

New in version 2.0. Label interval-class vector of every vertical moment in *expr*:

```

>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "alto" g4 f4""")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "bass" c,2""")
>>> score.append(staff)

>>> labeltools.label_vertical_moments_in_expr_with_interval_class_vectors(score)

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    _ \markup {
      \tiny
      0010020
    }
    e'8
    f'8
    _ \markup {

```

```

        \tiny
        1000020
    }
}
\new Staff {
    \clef "alto"
    g4
    f4
    _ \markup {
        \tiny
        0100110
    }
}
\new Staff {
    \clef "bass"
    c,2
    _ \markup {
        \tiny
        1000020
    }
}
>>

```

Return none.

labeltools.label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes

`labeltools.label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes` (*expr*, *markup_direction*)

New in version 2.0. Label pitch-classes of every vertical moment in *expr*:

```

>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "alto" g4 f4""")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "bass" c,2""")
>>> score.append(staff)

>>> labeltools.label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes(
...     score)

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    _ \markup {
      \small
      \column
      {
        7
        2
        0
      }
    }
  }

```

```

        }
    e'8
    f'8
    - \markup {
        \small
        \column
        {
            5
            0
        }
    }
}
\new Staff {
    \clef "alto"
    g4
    f4
    - \markup {
        \small
        \column
        {
            5
            4
            0
        }
    }
}
\new Staff {
    \clef "bass"
    c,2
    - \markup {
        \small
        \column
        {
            7
            0
        }
    }
}
>>

```

Return none.

labeltools.label_vertical_moments_in_expr_with_pitch_numbers

`labeltools.label_vertical_moments_in_expr_with_pitch_numbers` (*expr*,
markup_direction=Down)

New in version 2.0. Label pitch numbers of every vertical moment in *expr*:

```

>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "alto" g4 f4""")
>>> score.append(staff)
>>> staff = Staff(r"""\clef "bass" c,2""")
>>> score.append(staff)

```

```
>>> labeltools.label_vertical_moments_in_expr_with_pitch_numbers(score)
```

```
>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    - \markup {
      \small
      \column
      {
        2
        -5
        -24
      }
    }
    e'8
    f'8
    - \markup {
      \small
      \column
      {
        5
        -7
        -24
      }
    }
  }
  \new Staff {
    \clef "alto"
    g4
    f4
    - \markup {
      \small
      \column
      {
        4
        -7
        -24
      }
    }
  }
  \new Staff {
    \clef "bass"
    c,2
    - \markup {
      \small
      \column
      {
        0
        -5
        -24
      }
    }
  }
}
>>
```

Return none.

labeltools.remove_markup_from_leaves_in_expr

labeltools.remove_markup_from_leaves_in_expr(*expr*)

New in version 1.1. Remove markup from leaves in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> labeltools.label_leaves_in_expr_with_pitch_class_numbers(staff)
>>> f(staff)
\new Staff {
  c'8 _ \markup { \small 0 }
  d'8 _ \markup { \small 2 }
  e'8 _ \markup { \small 4 }
  f'8 _ \markup { \small 5 }
}

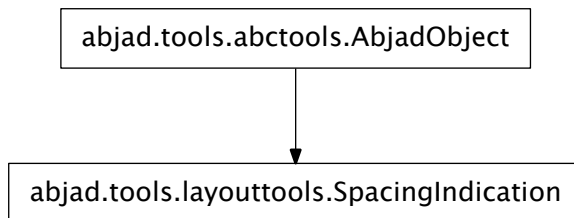
>>> labeltools.remove_markup_from_leaves_in_expr(staff)
>>> f(staff)
\new Staff {
  c'8
  d'8
  e'8
  f'8
}
```

Return `None`. Changed in version 2.0: renamed `label.clear_leaves()` to `labeltools.remove_markup_from_leaves_in_expr()`.

layouttools

concrete classes

layouttools.SpacingIndication



class layouttools.SpacingIndication(*args)

Spacing indication token.

LilyPond Score.proportionalNotationDuration will equal proportional_notation_duration when tempo equals tempo_indication:

```
>>> from abjad.tools import layouttools
```

Initialize from tempo mark and proportional notation duration:

```
>>> tempo = contexttools.TempoMark(Duration(1, 8), 44)
>>> indication = layouttools.SpacingIndication(tempo, Duration(1, 68))

>>> indication
SpacingIndication(TempoMark(Duration(1, 8), 44), Duration(1, 68))
```

Initialize from constants:

```
>>> layouttools.SpacingIndication(((1, 8), 44), (1, 68))
SpacingIndication(TempoMark(Duration(1, 8), 44), Duration(1, 68))
```

Initialize from other spacing indication:

```
>>> layouttools.SpacingIndication(indication)
SpacingIndication(TempoMark(Duration(1, 8), 44), Duration(1, 68))
```

Spacing indications are immutable.

Read-only Properties

`SpacingIndication.normalized_spacing_duration`

Read-only proportional notation duration at 60 MM.

`SpacingIndication.proportional_notation_duration`

LilyPond proportional notation duration context setting.

`SpacingIndication.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`SpacingIndication.tempo_indication`

Abjad tempo indication object.

Special Methods

`SpacingIndication.__eq__(expr)`

Spacing indications compare equal when normalized spacing durations compare equal.

`SpacingIndication.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SpacingIndication.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SpacingIndication.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SpacingIndication.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SpacingIndication.__ne__(expr)`

Spacing indications compare unequal when normalized spacing durations compare unequal.

`SpacingIndication.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

functions

`layouttools.make_spacing_vector`

`layouttools.make_spacing_vector(basic_distance, minimum_distance, padding, stretchability)`

New in version 2.0. Make spacing vector:

```
>>> from abjad.tools import layouttools
```

```
>>> vector = layouttools.make_spacing_vector(0, 0, 12, 0)
```

Use to set paper block spacing attributes:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> lilypond_file = lilypondfiletools.make_basic_lilypond_file(staff)
```

```
>>> spacing_vector = layouttools.make_spacing_vector(0, 0, 12, 0)
```

```
>>> lilypond_file.paper_block.system_system_spacing = spacing_vector
```

```
>>> f(lilypond_file)
```

```
% Abjad revision 4229
```

```
% 2011-04-07 15:19
```

```
\version "2.13.44"
```

```
\include "english.ly"
```

```
\include "/abjad/trunk/abjad/cfg/abjad.scm"
```

```
\paper {
```

```
  system-system-spacing = #'(
```

```
    (basic_distance . 0) (minimum_distance . 0) (padding . 12) (stretchability . 0))
```

```
}
```

```
\score {
```

```
  \new Staff {
```

```
    c'8
```

```
    d'8
```

```
    e'8
```

```
    f'8
```

```
  }
```

```
}
```

Return scheme vector.

layouttools.set_line_breaks_cyclically_by_line_duration_ge

`layouttools.set_line_breaks_cyclically_by_line_duration_ge`(*expr*, *line_duration*, *klass=None*, *add_empty_bars=False*)

Iterate *klass* instances in *expr* and accumulate prolated duration. Add line break after every total less than or equal to *line_duration*:

```
>>> from abjad.tools import layouttools

>>> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 4)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(t)

>>> f(t)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
  {
    b'8
    c''8
  }
}

>>> layouttools.set_line_breaks_cyclically_by_line_duration_ge(t, Duration(4, 8))
>>> f(t)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
    \break
  }
  {
    g'8
    a'8
  }
  {
    b'8
    c''8
    \break
  }
}
```

When `klass=None` set `klass` to measure.

Set `adjust_eol` to `True` to include a magic Scheme incantation to move end-of-line LilyPond TimeSignature and BarLine grobs to the right. Changed in version 2.0: renamed `layout.line_break_every_prolated()` to `layout.set_line_breaks_cyclically_by_line_duration_ge()`.

`layouttools.set_line_breaks_cyclically_by_line_duration_in_seconds_ge`

```
layouttools.set_line_breaks_cyclically_by_line_duration_in_seconds_ge(expr,
                                                                    line_duration,
                                                                    klass=None,
                                                                    ad-
                                                                    just_eol=False,
                                                                    add_empty_bars=False)
```

Iterate `klass` instances in `expr` and accumulate duration in seconds. Add line break after every total less than or equal to `line_duration`:

```
>>> from abjad.tools import layouttools

>>> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 4)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(t)
>>> tempo_mark = contexttools.TempoMark(Duration(1, 8), 44, target_context = Staff)(t)

>>> f(t)
\new Staff {
  \tempo 8=44
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
  {
    b'8
    c''8
  }
}

>>> layouttools.set_line_breaks_cyclically_by_line_duration_in_seconds_ge(t, Duration(6))
>>> f(t)
\new Staff {
  \tempo 8=44
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
```

```

        \break
    }
    {
        g'8
        a'8
    }
    {
        b'8
        c''8
    }
}

```

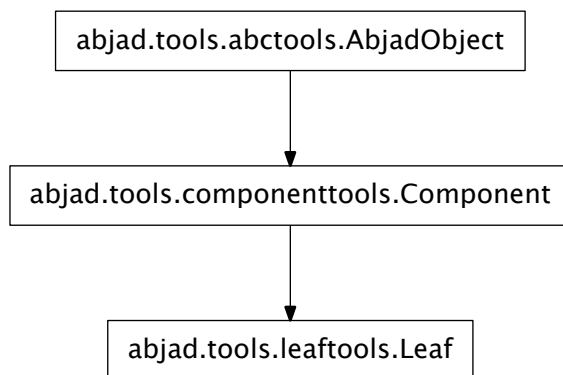
When `klass=None` set *klass* to measure.

Set `adjust_eol = True` to include a magic Scheme incantation to move end-of-line LilyPond TimeSignature and BarLine grobs to the right. Changed in version 2.0: renamed `layout.line_break_every_seconds()` to `layout.set_line_breaks_cyclically_by_line_duration_in_seconds_ge()`.

leaftools

concrete classes

leaftools.Leaf



```
class leaftools.Leaf (written_duration, duration_multiplier=None)
```

Read-only Properties

`Leaf.duration_in_seconds`

`Leaf.leaf_index`

`Leaf.lilypond_format`

Inherited from `componenttools.Component`

`Leaf.multiplied_duration`

Leaf.override
 Read-only reference to LilyPond grob override component plug-in.
 Inherited from `componenttools.Component`

Leaf.parent
 Inherited from `componenttools.Component`

Leaf.preprolated_duration

Leaf.prolated_duration
 Inherited from `componenttools.Component`

Leaf.prolation
 Inherited from `componenttools.Component`

Leaf.set
 Read-only reference LilyPond context setting component plug-in.
 Inherited from `componenttools.Component`

Leaf.spanners
 Read-only reference to unordered set of spanners attached to component.
 Inherited from `componenttools.Component`

Leaf.start_offset
 Read-only start offset of component.
 Inherited from `componenttools.Component`

Leaf.start_offset_in_seconds
 Read-only start offset of comonent in seconds.
 Inherited from `componenttools.Component`

Leaf.stop_offset
 Read-only stop offset of component.
 Inherited from `componenttools.Component`

Leaf.stop_offset_in_seconds
 Read-only stop offset of component in seconds.
 Inherited from `componenttools.Component`

Leaf.storage_format
 Storage format of Abjad object.
 Return string.
 Inherited from `abctools.AbjadObject`

Read/write Properties

Leaf.duration_multiplier

Leaf.written_duration

Leaf.written_pitch_indication_is_at_sounding_pitch

Leaf.written_pitch_indication_is_nonsemantic

Special Methods

Leaf.**__and__**(arg)

Leaf.**__eq__**(arg)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Leaf.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Leaf.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Leaf.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Leaf.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Leaf.**__mul__**(n)

Inherited from `componenttools.Component`

Leaf.**__ne__**(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Leaf.**__or__**(arg)

Leaf.**__repr__**()

Leaf.**__rmul__**(n)

Inherited from `componenttools.Component`

Leaf.**__str__**()

Leaf.**__sub__**(arg)

Leaf.**__xor__**(arg)

functions

leaftools.all_are_leaves

`leaftools.all_are_leaves(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad leaves:

```
>>> leaves = [Note("c'4"), Rest('r4'), Note("d'4")]
```

```
>>> leaftools.all_are_leaves(leaves)
True
```

True when *expr* is an empty sequence:

```
>>> leaftools.all_are_leaves([])
True
```

Otherwise false:

```
>>> leaftools.all_are_leaves('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_duration

`leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_duration(leaf, written_duration, preprolated_duration)`

New in version 1.1. Change *leaf* written duration to *written_duration* and preserve preprolated *leaf* duration:

```
>>> note = Note("c'4")
```

```
>>> note.written_duration
Duration(1, 4)
```

```
>>> note.preprolated_duration
Duration(1, 4)
```

```
>>> leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_duration(
...     note, Duration(3, 16))
Note("c'8. * 4/3")
```

```
>>> note.written_duration
Duration(3, 16)
```

```
>>> note.preprolated_duration
Duration(1, 4)
```

Add LilyPond multiplier where necessary.

Return *leaf*. Changed in version 2.0: Renamed from `leaftools.duration_rewrite()`.
`leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_duration()`.

leaftools.copy_written_duration_and_multiplier_from_leaf_to_leaf

`leaftools.copy_written_duration_and_multiplier_from_leaf_to_leaf` (*source_leaf*,
target_leaf)

New in version 2.0. Copy written duration and multiplier from *source_leaf* to *target_leaf*:

```
>>> note = Note("c'4")
>>> note.duration_multiplier = Duration(1, 2)
>>> rest = Rest((1, 64))
>>> leaftools.copy_written_duration_and_multiplier_from_leaf_to_leaf(note, rest)
Rest('r4 * 1/2')
```

Return *target_leaf*.

leaftools.divide_leaf_meiotically

`leaftools.divide_leaf_meiotically` (*leaf*, *n*=2)

New in version 1.1. Divide *leaf* meiotically *n* times:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

>>> leaftools.divide_leaf_meiotically(staff[0], n=4)

>>> f(staff)
\new Staff {
    c'32 [
    c'32
    c'32
    c'32
    d'8
    e'8
    f'8 ]
}
```

Replace *leaf* with *n* new leaves.

Preserve parentage and spanners.

Allow divisions into only 1, 2, 4, 8, 16, ... and other nonnegative integer powers of 2.

Produce only leaves and never tuplets or other containers.

Return none.

leaftools.divide_leaves_in_expr_meiotically

`leaftools.divide_leaves_in_expr_meiotically` (*expr*, *n*=2)

New in version 1.1. Divide leaves meiotically in *expr* *n* times:


```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}

>>> leaftools.divide_leaves_in_expr_meiotically(staff[2:], n=4)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'32
  e'32
  e'32
  e'32
  f'32
  f'32
  f'32
  f'32 ]
}
```

Replace every leaf in *expr* with *n* new leaves.

Preserve parentage and spanners.

Allow divisions into only 1, 2, 4, 8, 16, ... and other nonnegative integer powers of 2.

Produce only leaves and never tuples or other containers.

Return `None`. Changed in version 2.0: renamed `leaftools.meiose()` to `leaftools.divide_leaves_in_expr_meiotically()`.

leaftools.expr_has_leaf_with_dotted_written_duration

`leaftools.expr_has_leaf_with_dotted_written_duration(expr)`

New in version 2.0. True when *expr* has at least one leaf with dotted written duration:

```
>>> notes = notetools.make_notes([0], [(1, 16), (2, 16), (3, 16)])
>>> leaftools.expr_has_leaf_with_dotted_written_duration(notes)
True
```

False otherwise:

```
>>> notes = notetools.make_notes([0], [(1, 16), (2, 16), (4, 16)])
>>> leaftools.expr_has_leaf_with_dotted_written_duration(notes)
False
```

Return boolean.

leaftools.fuse_leaves

`leaftools.fuse_leaves` (*leaves*)

New in version 1.1. Fuse thread-contiguous *leaves*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> leaftools.fuse_leaves(staff[1:])
[Note("d'4.") ]
>>> f(staff)
\new Staff {
    c'8
    d'4.
}
```

Rewrite duration of first leaf in *leaves*.

Detach all leaves in *leaves* other than first leaf from score.

Return list of first leaf in *leaves*. Changed in version 2.0: renamed `fuse.leaves_by_reference()` to `leaftools.fuse_leaves()`.

leaftools.fuse_leaves_in_container_once_by_counts

`leaftools.fuse_leaves_in_container_once_by_counts` (*container*, *counts*, *klass=None*, *big_endian=True*)

Fuse leaves in *container* once by *counts* into instances of *klass*.

leaftools.fuse_leaves_in_tie_chain_by_immediate_parent

`leaftools.fuse_leaves_in_tie_chain_by_immediate_parent` (*tie_chain*)

New in version 1.1. Fuse leaves in *tie_chain* by immediate parent:

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> tietools.TieSpanner(staff.leaves)
TieSpanner(c'8, c'8, c'8, c'8)
>>> f(staff)
\new Staff {
    {
        \time 2/8
        c'8 ~
        c'8 ~
    }
    {
        c'8 ~
        c'8
    }
}

>>> tie_chain = tietools.get_tie_chain(staff.leaves[0])
>>> leaftools.fuse_leaves_in_tie_chain_by_immediate_parent(tie_chain)
[[Note("c'4")], [Note("c'4")]]

>>> f(staff)
\new Staff {
    {
        \time 2/8
```

```

        c'4 ~
    }
    {
        c'4
    }
}

```

Return list of fused notes by parent. Changed in version 2.0: renamed `fuse.leaves_in_tie_chain()` to `leaftools.fuse_leaves_in_tie_chain_by_immediate_parent()`.

`leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_without_overhang`

`leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_without_overhang` (*components*, *prolated_durations*)

New in version 1.1. Fuse tied leaves in *components* once by *prolated_durations* without overhang:

```

>>> staff = Staff(notetools.make_repeated_notes(8))
>>> tietools.TieSpanner(staff.leaves)
TieSpanner(c'8, c'8, c'8, c'8, c'8, c'8, c'8, c'8)

>>> f(staff)
\new Staff {
    c'8 ~
    c'8 ~
    c'8 ~
    c'8 ~
    c'8 ~
    c'8 ~
    c'8 ~
    c'8
}

>>> leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_without_overhang(
... staff, [Duration(3, 8), Duration(3, 8)])

>>> f(staff)
\new Staff {
    c'4. ~
    c'4. ~
    c'8 ~
    c'8
}

```

Return none. Changed in version 2.0: renamed `fuse.tied_leaves_by_prolated_durations()` to `leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_without_overhang()`.

`leaftools.get_composite_offset_difference_series_from_leaves_in_expr`

`leaftools.get_composite_offset_difference_series_from_leaves_in_expr` (*expr*)

New in version 2.0. Get composite offset difference series from leaves in *expr*:

```

>>> staff_1 = Staff(r"\times 4/3 { c'8 d'8 e'8 }")
>>> staff_2 = Staff("f'8 g'8 a'8 b'8")
>>> score = Score([staff_1, staff_2])

```

```
>>> f(score)
\new Score <<
  \new Staff {
    \fraction \times 4/3 {
      c'8
      d'8
      e'8
    }
  }
  \new Staff {
    f'8
    g'8
    a'8
    b'8
  }
>>

>>> for x in leaftools.get_composite_offset_difference_series_from_leaves_in_expr(score):
...     x
...
Duration(1, 8)
Duration(1, 24)
Duration(1, 12)
Duration(1, 12)
Duration(1, 24)
Duration(1, 8)
```

Composite offset difference series defined equal to time intervals between unique start and stop offsets of leaves in *expr*.

Return list of durations.

`leaftools.get_composite_offset_series_from_leaves_in_expr`

`leaftools.get_composite_offset_series_from_leaves_in_expr(expr)`

New in version 2.0. Get composite offset series from leaves in *expr*:

```
>>> staff_1 = Staff(r"\times 4/3 { c'8 d'8 e'8 }")
>>> staff_2 = Staff("f'8 g'8 a'8 b'8")
>>> score = Score([staff_1, staff_2])

>>> f(score)
\new Score <<
  \new Staff {
    \fraction \times 4/3 {
      c'8
      d'8
      e'8
    }
  }
  \new Staff {
    f'8
    g'8
    a'8
    b'8
  }
>>
```

```
>>> for offset in leaftools.get_composite_offset_series_from_leaves_in_expr(score):
...     offset
...
Offset(0, 1)
Offset(1, 8)
Offset(1, 6)
Offset(1, 4)
Offset(1, 3)
Offset(3, 8)
Offset(1, 2)
```

Equal to list of unique start and stop offsets of leaves in *expr*.

Return list of fractions.

leaftools.get_leaf_at_index_in_measure_number_in_expr

`leaftools.get_leaf_at_index_in_measure_number_in_expr(expr, measure_number, leaf_index)`

New in version 2.0. Get leaf at *leaf_index* in *measure_number* in *expr*:

```
>>> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(t)
>>> f(t)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
}

>>> leaftools.get_leaf_at_index_in_measure_number_in_expr(t, 2, 0)
Note("e'8")
```

Return leaf or none.

leaftools.get_leaf_in_expr_with_maximum_prolated_duration

`leaftools.get_leaf_in_expr_with_maximum_prolated_duration(expr)`

New in version 2.5. Get leaf in *expr* with maximum prolated duration:

```
>>> staff = Staff("c'4.. d'16 e'4.. f'16")

>>> leaftools.get_leaf_in_expr_with_maximum_prolated_duration(staff)
Note("c'4..")
```

When two leaves in *expr* are both of equally maximal prolated duration, return the first leaf encountered in forward iteration.

Return none when *expr* contains no leaves:

```
>>> leaftools.get_leaf_in_expr_with_maximum_prolated_duration([]) is None
True
```

Return leaf.

leaftools.get_leaf_in_expr_with_minimum_prolated_duration

`leaftools.get_leaf_in_expr_with_minimum_prolated_duration(expr)`

New in version 2.5. Get leaf in *expr* with minimum prolated duration:

```
>>> staff = Staff("c'4.. d'16 e'4.. f'16")

>>> leaftools.get_leaf_in_expr_with_minimum_prolated_duration(staff)
Note("d'16")
```

When two leaves in *expr* are both of equally minimal prolated duration, return the first leaf encountered in forward iteration.

Return none when *expr* contains no leaves:

```
>>> leaftools.get_leaf_in_expr_with_minimum_prolated_duration([]) is None
True
```

Return leaf.

leaftools.get_nth_leaf_in_expr

`leaftools.get_nth_leaf_in_expr(expr, n=0)`

New in version 2.0. Get *n* th leaf in *expr*:

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
}

>>> for n in range(6):
...     leaftools.get_nth_leaf_in_expr(staff, n)
...
Note("c'8")
Note("d'8")
Note("e'8")
```

```
Note("f'8")
Note("g'8")
Note("a'8")
```

Read backwards for negative values of n .

```
>>> leaftools.get_nth_leaf_in_expr(staff, -1)
Note("a'8")
```

Note: Because this function returns as soon as it finds instance n of *klases*, it is more efficient to call `leaftools.get_nth_leaf_in_expr(expr, 0)` than `expr.leaves[0]`. It is likewise more efficient to call `leaftools.get_nth_leaf_in_expr(expr, -1)` than `expr.leaves[-1]`.

Return leaf of none.

leaftools.get_nth_leaf_in_thread_from_leaf

`leaftools.get_nth_leaf_in_thread_from_leaf(leaf, n=0)`

New in version 2.0. Get n th leaf in thread from *leaf*:

```
>>> staff = Staff(2 * Voice("c'8 d'8 e'8 f'8"))
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> f(staff)
\new Staff {
  \new Voice {
    c'8
    d'8
    e'8
    f'8
  }
  \new Voice {
    g'8
    a'8
    b'8
    c''8
  }
}

>>> for n in range(8):
...     print n, leaftools.get_nth_leaf_in_thread_from_leaf(staff[0][0], n)
...
0 c'8
1 d'8
2 e'8
3 f'8
4 None
5 None
6 None
7 None
```

Return leaf or none.

leaftools.is_bar_line_crossing_leaf

`leaftools.is_bar_line_crossing_leaf(leaf)`

New in version 2.0. True when *leaf* crosses bar line:

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> t[2].written_duration *= 2
>>> contexttools.TimeSignatureMark((2, 8), partial = Duration(1, 8))(t[2])
TimeSignatureMark((2, 8), partial=Duration(1, 8))(e'4)
>>> f(t)
\new Staff {
    c'8
    d'8
    \partial 8
    \time 2/8
    e'4
    f'8
}
>>> leaftools.is_bar_line_crossing_leaf(t.leaves[2])
True
```

Otherwise false:

```
>>> leaftools.is_bar_line_crossing_leaf(t.leaves[3])
False
```

Return boolean.

leaftools.list_prolated_durations_of_leaves_in_expr

`leaftools.list_prolated_durations_of_leaves_in_expr(expr)`

New in version 2.0. List prolated durations of leaves in *expr*:

```
>>> staff = Staff(r"      imes 2/3 { c'8 d'8 e'8 }      imes 2/3 { c'8 d'8 e'8 }")
>>> for duration in leaftools.list_prolated_durations_of_leaves_in_expr(staff):
...     duration
...
Duration(1, 12)
Duration(1, 12)
Duration(1, 12)
Duration(1, 12)
Duration(1, 12)
Duration(1, 12)
```

Return list of durations.

leaftools.list_written_durations_of_leaves_in_expr

`leaftools.list_written_durations_of_leaves_in_expr(expr)`

New in version 2.0. List the written durations of leaves in *expr*:

```
>>> staff = Staff(tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8") * 2)
>>> for duration in leaftools.list_written_durations_of_leaves_in_expr(staff):
...     duration
```



```
...
Duration(1, 8)
Duration(1, 8)
Duration(1, 8)
Duration(1, 8)
Duration(1, 8)
Duration(1, 8)
```

Return list of durations.

leaftools.make_leaves

`leaftools.make_leaves(pitches, durations, big_endian=True, tie_rests=False)`

New in version 1.1. Construct a list of notes, rests or chords.

Set *pitches* is a single pitch, a list of pitches, or a tuple of pitches.

Integer pitches create notes.

```
>>> leaftools.make_leaves([2, 4, 19], [(1, 4)])
[Note("d'4"), Note("e'4"), Note("g'4")]
```

Tuple pitches create chords.

```
>>> leaftools.make_leaves([(0, 1, 2), (3, 4, 5), (6, 7, 8)], [(1, 4)])
[Chord("<c' cs' d'>4"), Chord("<ef' e' f'>4"), Chord("<fs' g' af'>4")]
```

Set *pitches* to a list of none to create rests.

```
>>> leaftools.make_leaves([None, None, None, None], [(1, 8)])
[Rest('r8'), Rest('r8'), Rest('r8'), Rest('r8')]
```

You can mix and match pitch values.

```
>>> leaftools.make_leaves([12, (1, 2, 3), None, 12], [(1, 4)])
[Note("c''4"), Chord("<cs' d' ef'>4"), Rest('r4'), Note("c''4")]
```

If the length of *pitches* is less than the length of *durations*, the function reads *durations* cyclically.

```
>>> leaftools.make_leaves([13], [(1, 8), (1, 8), (1, 4), (1, 4)])
[Note("cs''8"), Note("cs''8"), Note("cs''4"), Note("cs''4")]
```

Set *durations* to a single duration, a list of duration, or a tuple of durations.

If the length of *durations* is less than the length of *pitches*, the function reads *pitches* cyclically.

```
>>> leaftools.make_leaves([13, 14, 15, 16], [(1, 8)])
[Note("cs''8"), Note("d''8"), Note("ef''8"), Note("e''8")]
```

Duration values not of the form $m / 2 ** n$ return leaves nested inside a fixed-multiplier tuplet.

```
>>> leaftools.make_leaves([14], [(1, 12), (1, 12), (1, 12)])
[Tuplet(2/3, [d''8, d''8, d''8])]
```

Set `big_endian=False` to return tied leaf durations from least to greatest:

```
>>> staff = Staff(leaftools.make_leaves([15], [(13, 16)], big_endian=False))
>>> f(staff)
\new Staff {
    ef''16 ~
```

```
    ef''2.
}
```

Set `tie_rests` to true to return tied rests for durations like 5/16 and 9/16.

```
>>> staff = Staff(leaftools.make_leaves([None], [(5, 16)], tie_rests=True))
>>> f(staff)
\new Staff {
    r4 ~
    r16
}
```

Return list of leaves. Changed in version 2.0: renamed `construct.leaves()` to `leaftools.make_leaves()`.

leaftools.make_leaves_from_note_value_signal

`leaftools.make_leaves_from_note_value_signal` (*note_value_signal*, *denominator_of_signal*, *tie_rests=False*, *use_skips=False*)

New in version 2.0. Make leaves from *note_value_signal* and *denominator_of_signal*:

```
>>> leaves = leaftools.make_leaves_from_note_value_signal([3, -3, 5, -5], 8)
>>> staff = Staff(leaves)

>>> f(staff)
\new Staff {
    c'4.
    r4.
    c'2 ~
    c'8
    r2
    r8
}
```

Interpret positive elements in *note_value_signal* as notes.

Interpret negative elements in *note_value_signal* as rests.

Set the pitch of all notes to middle C.

When `use_skips=False` use skips instead of rests.

Note: Add skip example.

Return list of notes and / or rests.

leaftools.make_tied_leaf

`leaftools.make_tied_leaf` (*kind*, *duration*, *big_endian=True*, *pitches=None*, *tied=True*)

Return list of leaves to fill the given duration *duration*.

Leaves returned are tie-spanned when `tied=True`.

duration must be of the form `m / 2**n` for any integer `m`.

big_endian must be boolean. True returns a list of notes of decreasing duration. False returns a list of notes of increasing duration.

pitches a pitch or list of pitch tokens.

tied True to return tied leaves. False otherwise.

leaftools.remove_initial_rests_from_sequence

`leaftools.remove_initial_rests_from_sequence(sequence)`

New in version 2.0. Remove initial rests from *sequence*:

```
>>> staff = Staff("r8 r8 c'8 d'8 r4 r4")

>>> f(staff)
\new Staff {
    r8
    r8
    c'8
    d'8
    r4
    r4
}

>>> leaftools.remove_initial_rests_from_sequence(staff)
[Note("c'8"), Note("d'8"), Rest('r4'), Rest('r4')]

>>> f(staff)
\new Staff {
    r8
    r8
    c'8
    d'8
    r4
    r4
}
```

Return list.

leaftools.remove_leaf_and_shrink_durated_parent_containers

`leaftools.remove_leaf_and_shrink_durated_parent_containers(leaf)`

New in version 1.1. Remove *leaf* and shrink durated parent containers:

```
>>> measure = Measure((4, 8), [])
>>> measure.append(tuplettools.FixedDurationTuplet((2, 8), "c'8 d'8 e'8"))
>>> measure.append(tuplettools.FixedDurationTuplet((2, 8), "f'8 g'8 a'8"))
>>> beamtools.BeamSpanner(measure.leaves)
BeamSpanner(c'8, d'8, e'8, f'8, g'8, a'8)

>>> f(measure)
{
    \time 4/8
    \times 2/3 {
        c'8 [
        d'8
        e'8
        ]
    }
    \times 2/3 {
```

```

        f'8
        g'8
        a'8 ]
    }
}

>>> leaftools.remove_leaf_and_shrink_durated_parent_containers(measure.leaves[0])

>>> f(measure)
{
  \time 5/12
  \scaleDurations #'(2 . 3) {
    {
      d'8 [
      e'8
    ]
    {
      f'8
      g'8
      a'8 ]
    }
  }
}

```

Return none.

leaftools.remove_outer_rests_from_sequence

`leaftools.remove_outer_rests_from_sequence(sequence)`

New in version 2.0. Remove outer rests from *sequence*:

```

>>> staff = Staff("r8 r8 c'8 d'8 r4 r4")

>>> f(staff)
\new Staff {
  r8
  r8
  c'8
  d'8
  r4
  r4
}

>>> leaftools.remove_outer_rests_from_sequence(staff)
[Note("c'8"), Note("d'8")]

>>> f(staff)
\new Staff {
  r8
  r8
  c'8
  d'8
  r4
  r4
}

```

Return list.

leaftools.remove_terminal_rests_from_sequence

`leaftools.remove_terminal_rests_from_sequence(sequence)`

New in version 2.0. Remove terminal rests from *sequence*:

```
>>> staff = Staff("r8 r8 c'8 d'8 r4 r4")

>>> f(staff)
\new Staff {
    r8
    r8
    c'8
    d'8
    r4
    r4
}

>>> leaftools.remove_terminal_rests_from_sequence(staff)
[Rest('r8'), Rest('r8'), Note("c'8"), Note("d'8")]

>>> f(staff)
\new Staff {
    r8
    r8
    c'8
    d'8
    r4
    r4
}
```

Return list.

leaftools.repeat_leaf

`leaftools.repeat_leaf(leaf, total=1)`

New in version 1.1. Repeat *leaf* and extend spanners:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

>>> leaftools.repeat_leaf(staff[0], total=3)

>>> f(staff)
\new Staff {
    c'8 [
    c'8
    c'8
    d'8
    e'8
```

```
f'8 ]
}
```

Preserve *leaf* written duration.

Preserve parentage and spanners.

Return none. Changed in version 2.0: renamed `leaftools.clone_and_splice_leaf()` to `leaftools.repeat_leaf()`.

`leaftools.repeat_leaves_in_expr`

`leaftools.repeat_leaves_in_expr(expr, total=1)`

New in version 1.1. Repeat leaves in *expr* and extend spanners:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}

>>> result = leaftools.repeat_leaves_in_expr(staff[2:], total=3)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  e'8
  e'8
  f'8
  f'8
  f'8 ]
}
```

Preserve leaf written durations.

Preserve parentage and spanners.

Return none. Changed in version 2.0: renamed `leaftools.multiply()` to `leaftools.repeat_leaves_in_expr()`.

`leaftools.replace_leaves_in_expr_with_named_parallel_voices`

`leaftools.replace_leaves_in_expr_with_named_parallel_voices(expr, upper_name, lower_name)`

Replace leaves in *expr* with two parallel voices containing copies of leaves in *expr*, with the upper voice named *upper_name* and the lower voice named *lower_name*:

```
>>> c = p('{ c c c c }')
>>> f(c)
{
```

```

    c4
    c4
    c4
    c4
}

>>> leaves = leaftools.replace_leaves_in_expr_with_named_parallel_voices(
... c.leaves[1:3], 'upper', 'lower')

>>> f(c)
{
  c4
  <<
    \context Voice = "upper" {
      c4
      c4
    }
    \context Voice = "lower" {
      c4
      c4
    }
  >>
  c4
}

```

If leaves in *expr* have different immediate parents, parallel voices will be created in each parent:

```

>>> c = p(r'{ c8 \times 2/3 { c8 c c } \times 4/5 { c16 c c c c } c8 }')
>>> f(c)
{
  c8
  \times 2/3 {
    c8
    c8
    c8
  }
  \times 4/5 {
    c16
    c16
    c16
    c16
    c16
  }
  c8
}

>>> leaves = leaftools.replace_leaves_in_expr_with_named_parallel_voices(
... c.leaves[2:7], 'upper', 'lower')

>>> f(c)
{
  c8
  \times 2/3 {
    c8
    <<
      \context Voice = "upper" {
        c8
        c8
      }

```

```

        \context Voice = "lower" {
            c8
            c8
        }
    >>
}
\times 4/5 {
    <<
        \context Voice = "upper" {
            c16
            c16
            c16
        }
        \context Voice = "lower" {
            c16
            c16
            c16
        }
    >>
    c16
    c16
}
c8
}

```

Returns a list leaves in upper voice, and a list of leaves in lower voice.

leaftools.replace_leaves_in_expr_with_parallel_voices

leaftools.replace_leaves_in_expr_with_parallel_voices (*expr*)

Replace leaves in *expr* with two parallel voices containing copies of leaves in *expr*:

```

>>> c = p('{ c c c c }')
>>> f(c)
{
    c4
    c4
    c4
    c4
}

>>> leaftools.replace_leaves_in_expr_with_parallel_voices(c.leaves[1:3])
([Note('c4'), Note('c4')], [Note('c4'), Note('c4')])

>>> f(c)
{
    c4
    <<
        \new Voice {
            c4
            c4
        }
        \new Voice {
            c4
            c4
        }
    >>
}

```



```

    c4
}

```

If leaves in *expr* have different immediate parents, parallel voices will be created in each parent:

```

>>> c = p(r'{ c8 \times 2/3 { c8 c c } \times 4/5 { c16 c c c c } c8 }')
>>> f(c)
{
  c8
  \times 2/3 {
    c8
    c8
    c8
  }
  \times 4/5 {
    c16
    c16
    c16
    c16
    c16
  }
  c8
}

>>> leaves = leaftools.replace_leaves_in_expr_with_parallel_voices(c.leaves[2:7])

>>> f(c)
{
  c8
  \times 2/3 {
    c8
    <<
      \new Voice {
        c8
        c8
      }
      \new Voice {
        c8
        c8
      }
    >>
  }
  \times 4/5 {
    <<
      \new Voice {
        c16
        c16
        c16
      }
      \new Voice {
        c16
        c16
        c16
      }
    >>
    c16
    c16
  }
  c8
}

```

```
}
```

Returns a list leaves in upper voice, and a list of leaves in lower voice.

leaftools.rest_leaf_at_offset

leaftools.rest_leaf_at_offset (*leaf*, *offset*)

New in version 1.1. Split *leaf* at *offset* and rest right half:

```
>>> staff = Staff("c'8 ( d'8 e'8 f'8 )")

>>> f(staff)
\new Staff {
  c'8 (
    d'8
    e'8
    f'8 )
}

>>> leaftools.rest_leaf_at_offset(staff.leaves[1], (1, 32))
([Note("d'32")], [Note("d'16.")])

>>> f(staff)
\new Staff {
  c'8 (
    d'32
    r16.
    e'8
    f'8 )
}
```

Return list of leaves to left of *offset* together with list of leaves to right of *offset*. Changed in version 2.0: renamed `leaftools.shorten()` to `leaftools.rest_leaf_at_offset()`.

leaftools.scale_preprolated_leaf_duration

leaftools.scale_preprolated_leaf_duration (*leaf*, *multiplier*)

New in version 1.1. Scale preprolated *leaf* leaf duration by dotted *multiplier*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> leaftools.scale_preprolated_leaf_duration(staff[1], Duration(3, 2))
[Note("d'8.") ]
>>> f(staff)
\new Staff {
  c'8 [
    d'8.
    e'8
    f'8 ]
}
```

Scale preprolated *leaf* duration by tied *multiplier*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
```

```

BeamSpanner(c'8, d'8, e'8, f'8)
>>> leaftools.scale_preprolated_leaf_duration(staff[1], Duration(5, 4))
[Note("d'8"), Note("d'32")]
>>> f(staff)
\new Staff {
    c'8 [
    d'8 ~
    d'32
    e'8
    f'8 ]
}

```

Scale preprolated *leaf* duration by nonbinary *multiplier*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> leaftools.scale_preprolated_leaf_duration(staff[1], Duration(2, 3))
[Note("d'8")]
>>> f(staff)
\new Staff {
    c'8 [
    \times 2/3 {
        d'8
    }
    e'8
    f'8 ]
}

```

Scale preprolated *leaf* duration by tied nonbinary *multiplier*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> leaftools.scale_preprolated_leaf_duration(staff[1], Duration(5, 6))
[Note("d'8"), Note("d'32")]
>>> f(staff)
\new Staff {
    c'8 [
    \times 2/3 {
        d'8 ~
        d'32
    }
    e'8
    f'8 ]
}

```

Return *leaf*. Changed in version 2.0: renamed from `leaftools.duration_scale()`.
`leaftools.scale_preprolated_leaf_duration()`.

leaftools.set_preprolated_leaf_duration

`leaftools.set_preprolated_leaf_duration` (*leaf*, *new_preprolated_duration*)

New in version 1.1. Set preprolated *leaf* duration:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

```

```
>>> leaftools.set_preprolated_leaf_duration(staff[1], Duration(3, 16))
[Note("d'8.") ]
>>> f(staff)
\new Staff {
  c'8 [
  d'8.
  e'8
  f'8 ]
}
```

Set tied preprolated *leaf* duration:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> leaftools.set_preprolated_leaf_duration(staff[1], Duration(5, 32))
[Note("d'8"), Note("d'32")]
>>> f(staff)
\new Staff {
  c'8 [
  d'8 ~
  d'32
  e'8
  f'8 ]
}
```

Set nonbinary preprolated *leaf* duration:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> leaftools.set_preprolated_leaf_duration(staff[1], Duration(1, 12))
[Note("d'8")]
>>> f(staff)
\new Staff {
  c'8 [
  \times 2/3 {
    d'8
  }
  e'8
  f'8 ]
}
```

Set tied nonbinary preprolated *leaf* duration:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
>>> leaftools.set_preprolated_leaf_duration(staff[1], Duration(5, 48))
[Note("d'8"), Note("d'32")]
>>> f(staff)
\new Staff {
  c'8 [
  \times 2/3 {
    d'8 ~
    d'32
  }
  e'8
  f'8 ]
}
```

```
}
```

Set preprolated *leaf* duration with LilyPond multiplier:

```
>>> note = Note(0, (1, 8))
>>> note.duration_multiplier = Duration(1, 2)
>>> leaftools.set_preprolated_leaf_duration(note, Duration(5, 48))
[Note("c'8 * 5/6")]
>>> f(note)
c'8 * 5/6
```

Return list of *leaf* and leaves newly tied to *leaf*. Changed in version 2.0: renamed `leaftools.change_leaf_preprolated_duration()` to `leaftools.set_preprolated_leaf_duration()`.

leaftools.show_leaves

`leaftools.show_leaves` (*leaves*, *suppress_pdf=False*)

New in version 2.0. Show *leaves* in temporary piano staff score:

```
>>> leaves = leaftools.make_leaves([None, 1, (-24, -22, 7, 21), None], (1, 4))
>>> score = leaftools.show_leaves(leaves)
\new Score <<
  \new PianoStaff <<
    \context Staff = "treble" {
      \clef "treble"
      r4
      cs'4
      <g' a''>4
      r4
    }
    \context Staff = "bass" {
      \clef "bass"
      r4
      r4
      <c, d,>4
      r4
    }
  >>
>>
```

Useful when working with notes, rests, chords not yet added to score.

Return temporary piano staff score.

leaftools.split_leaf_at_offset

`leaftools.split_leaf_at_offset` (*leaf*, *offset*, *fracture_spanners=False*, *tie_split_notes=True*, *tie_split_rests=False*)

Split *leaf* at *offset*.

Example 1. Split note at assignable offset. Two notes result. Do not tie notes:

```
>>> staff = Staff(r"abj: | 2/8 c'8 ( d'8 || 2/8 e'8 f'8 ) |")
>>> beamtools.apply_beam_spanners_to_measures_in_expr(staff)
[BeamSpanner(|2/8(2)|), BeamSpanner(|2/8(2)|)]
>>> contexttools.DynamicMark('f')(staff.leaves[0])
```

```

DynamicMark('f')(c'8)
>>> marktools.Articulation('accent')(staff.leaves[0])
Articulation('accent')(c'8)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 -\accent \f [ (
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}

>>> leaftools.split_leaf_at_offset(staff.leaves[0], (1, 32), tie_split_notes=False)
([Note("c'32")], [Note("c'16.")])

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'32 -\accent \f [ (
    c'16.
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}

```

Example 2. Handle grace and after grace containers correctly.

```

>>> staff = Staff(r"abj: | 2/8 c'8 ( d'8 || 2/8 e'8 f'8 ) |")
>>> beamtools.apply_beam_spanners_to_measures_in_expr(staff)
[BeamSpanner(|2/8(2)|), BeamSpanner(|2/8(2)|)]
>>> gracetools.GraceContainer("cs'16")(staff.leaves[0])
Note("c'8")
>>> gracetools.GraceContainer("ds'16", kind='after')(staff.leaves[0])
Note("c'8")

>>> f(staff)
\new Staff {
  {
    \time 2/8
    \grace {
      cs'16
    }
    \afterGrace
    c'8 [ (
    {
      ds'16
    }
    d'8 ]
  }
}

```

```

        e'8 [
        f'8 ] )
    }
}

>>> leaftools.split_leaf_at_offset(staff.leaves[0], (1, 32), tie_split_notes=False)
([Note("c'32")], [Note("c'16.")])

>>> f(staff)
\new Staff {
  {
    \time 2/8
    \grace {
      cs'16
    }
    c'32 [ (
    \afterGrace
    c'16.
    {
      ds'16
    }
    d'8 ]
  }
  {
    e'8 [
    f'8 ] )
  }
}

```

Return pair.

leaftools.split_leaf_at_offsets

`leaftools.split_leaf_at_offsets`(*leaf*, *offsets*, *cyclic=False*, *fracture_spanners=False*, *tie_split_notes=True*, *tie_split_rests=False*)

New in version 2.10. Split *leaf* at *offsets*.

Example 1. Split note once at *offsets* and tie split notes:

```

>>> staff = Staff("c'1 ( d'1 )")

>>> f(staff)
\new Staff {
  c'1 (
  d'1 )
}

>>> leaftools.split_leaf_at_offsets(staff[0], [(3, 8)], tie_split_notes=True)
[[Note("c'4."), [Note("c'2"), Note("c'8")]]]

>>> f(staff)
\new Staff {
  c'4. ( ~
  c'2 ~
  c'8
  d'1 )
}

```

Example 2. Split note cyclically at *offsets* and tie split notes:

```
>>> staff = Staff("c'1 ( d'1 )")

>>> f(staff)
\new Staff {
  c'1 (
    d'1 )
}

>>> leaftools.split_leaf_at_offsets(staff[0], [(3, 8)], cyclic=True, tie_split_notes=True)
[[Note("c'4."), [Note("c'4."), [Note("c'4")]]]

>>> f(staff)
\new Staff {
  c'4. ( ~
  c'4. ~
  c'4
  d'1 )
}
```

Example 3. Split note once at *offsets* and do no tie split notes:

```
>>> staff = Staff("c'1 ( d'1 )")

>>> f(staff)
\new Staff {
  c'1 (
    d'1 )
}

>>> leaftools.split_leaf_at_offsets(staff[0], [(3, 8)], tie_split_notes=False)
[[Note("c'4."), [Note("c'2"), Note("c'8")]]]

>>> f(staff)
\new Staff {
  c'4. (
  c'2 ~
  c'8
  d'1 )
}
```

Example 4. Split note cyclically at *offsets* and do not tie split notes:

```
>>> staff = Staff("c'1 ( d'1 )")

>>> f(staff)
\new Staff {
  c'1 (
    d'1 )
}

>>> leaftools.split_leaf_at_offsets(staff[0], [(3, 8)], cyclic=True, tie_split_notes=False)
[[Note("c'4."), [Note("c'4."), [Note("c'4")]]]

>>> f(staff)
\new Staff {
  c'4. (
  c'4.
```



```

    c'4
    d'1 )
}

```

Example 5. Split tupletted note once at *offsets* and tie split notes:

```

>>> staff = Staff(r"\times 2/3 { c'2 ( d'2 e'2 ) }")

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'2 (
      d'2
      e'2 )
  }
}

>>> leaftools.split_leaf_at_offsets(staff.leaves[1], [(1, 6)], cyclic=False, tie_split_notes=True)
[[Note("d'4")], [Note("d'4")]]

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'2 (
      d'4 ~
      d'4
      e'2 )
  }
}

```

Note: Add examples showing mark and context mark handling.

Return list of shards.

leaftools.yield_groups_of_mixed_notes_and_chords_in_sequence

leaftools.yield_groups_of_mixed_notes_and_chords_in_sequence(*sequence*)

New in version 2.0. Yield groups of mixed notes and chords in *sequence*:

```

>>> staff = Staff("c'8 d'8 r8 r8 <e' g'>8 <f' a'>8 g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

>>> f(staff)
\new Staff {
  c'8
  d'8
  r8
  r8
  <e' g'>8
  <f' a'>8
  g'8
  a'8
  r8
  r8
  <b' d''>8
  <c'' e''>8
}

```

```
>>> for group in leaftools.yield_groups_of_mixed_notes_and_chords_in_sequence(staff):
...     group
...
(Note("c'8"), Note("d'8"))
(Chord("<e' g'>8"), Chord("<f' a'>8"), Note("g'8"), Note("a'8"))
(Chord("<b' d'>8"), Chord("<c' e'>8"))
```

Return generator.

lilypondfiletools

abstract classes

lilypondfiletools.AttributedBlock

abjad.tools.lilypondfiletools.AttributedBlock

class lilypondfiletools.**AttributedBlock**

New in version 2.0. Abjad model of LilyPond input file block with attributes.

Read-only Properties

AttributedBlock.**lilypond_format**

Read/write Properties

AttributedBlock.**is_formatted_when_empty**

Special Methods

AttributedBlock.**__repr__()**

lilypondfiletools.NonattributedBlock

abjad.tools.lilypondfiletools.NonattributedBlock

class lilypondfiletools.**NonattributedBlock**

New in version 2.0. Abjad model of LilyPond input file block with no attributes.

Read-only Properties

`NonattributedBlock.lilypond_format`

Read/write Properties

`NonattributedBlock.is_formatted_when_empty`

Methods

`NonattributedBlock.append()`

`L.append(object)` – append object to end

Inherited from `__builtin__.list`

`NonattributedBlock.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`NonattributedBlock.extend()`

`L.extend(iterable)` – extend list by appending elements from the iterable

Inherited from `__builtin__.list`

`NonattributedBlock.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`NonattributedBlock.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`NonattributedBlock.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`NonattributedBlock.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`NonattributedBlock.reverse()`

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`NonattributedBlock.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`NonattributedBlock.__add__()`

`x.__add__(y) <==> x+y`

Inherited from `__builtin__.list`

`NonattributedBlock.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.list`

`NonattributedBlock.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Inherited from `__builtin__.list`

`NonattributedBlock.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

`NonattributedBlock.__eq__()`
`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.list`

`NonattributedBlock.__ge__()`
`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.list`

`NonattributedBlock.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.list`

`NonattributedBlock.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

`NonattributedBlock.__gt__()`
`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.list`

`NonattributedBlock.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Inherited from `__builtin__.list`

`NonattributedBlock.__imul__()`
`x.__imul__(y) <==> x*=y`

Inherited from `__builtin__.list`

`NonattributedBlock.__iter__()` <==> `iter(x)`

Inherited from `__builtin__.list`

`NonattributedBlock.__le__()`
`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.list`

`NonattributedBlock.__len__()` <==> `len(x)`

Inherited from `__builtin__.list`

`NonattributedBlock.__lt__()`
`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.list`

```

NonattributedBlock.__mul__()
    x.__mul__(n) <==> x*n

    Inherited from __builtin__.list

NonattributedBlock.__ne__()
    x.__ne__(y) <==> x!=y

    Inherited from __builtin__.list

NonattributedBlock.__repr__()

NonattributedBlock.__reversed__()
    L.__reversed__() – return a reverse iterator over the list

    Inherited from __builtin__.list

NonattributedBlock.__rmul__()
    x.__rmul__(n) <==> n*x

    Inherited from __builtin__.list

NonattributedBlock.__setitem__()
    x.__setitem__(i, y) <==> x[i]=y

    Inherited from __builtin__.list

NonattributedBlock.__setslice__()
    x.__setslice__(i, j, y) <==> x[i:j]=y

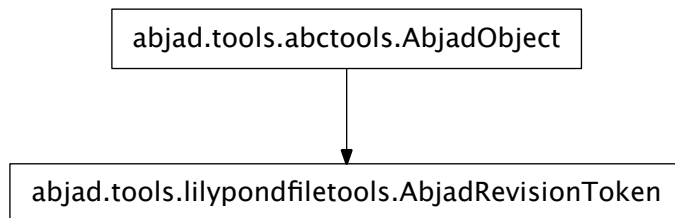
    Use of negative indices is not supported.

    Inherited from __builtin__.list

```

concrete classes

`lilypondfiletools.AbjadRevisionToken`



class `lilypondfiletools.AbjadRevisionToken`

New in version 2.0. Abjad version token:

```

>>> lilypondfiletools.AbjadRevisionToken()
AbjadRevisionToken(Abjad revision ...)

```

Return Abjad version token.

Read-only Properties

`AbjadRevisionToken.lilypond_format`

Format contribution of Abjad version token:

```
>>> lilypondfiletools.AbjadRevisionToken().lilypond_format
'Abjad revision ...'
```

Return string.

`AbjadRevisionToken.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`AbjadRevisionToken.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

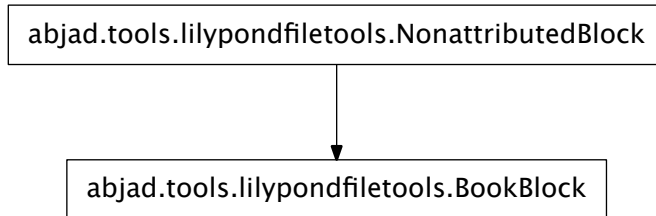
`AbjadRevisionToken.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__repr__()`

lilypondfiletools.BookBlock**class lilypondfiletools.BookBlock**

New in version 2.0. Abjad model of LilyPond input file book block:

```

>>> book_block = lilypondfiletools.BookBlock()

>>> book_block
BookBlock()

>>> f(book_block)
\book {}

```

Return book block.

Read-only Properties

BookBlock.lilypond_format

Inherited from `lilypondfiletools.NonattributedBlock`

Read/write Properties

BookBlock.is_formatted_when_empty

Inherited from `lilypondfiletools.NonattributedBlock`

Methods

BookBlock.append()

`L.append(object)` – append object to end

Inherited from `__builtin__.list`

BookBlock.count(value) → integer – return number of occurrences of value

Inherited from `__builtin__.list`

BookBlock.extend()

`L.extend(iterable)` – extend list by appending elements from the iterable

Inherited from `__builtin__.list`

BookBlock.index(value[, start[, stop]]) → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`BookBlock.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`BookBlock.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`BookBlock.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`BookBlock.reverse()`

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`BookBlock.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`BookBlock.__add__()`

`x.__add__(y)` <==> `x+y`

Inherited from `__builtin__.list`

`BookBlock.__contains__()`

`x.__contains__(y)` <==> `y in x`

Inherited from `__builtin__.list`

`BookBlock.__delitem__()`

`x.__delitem__(y)` <==> `del x[y]`

Inherited from `__builtin__.list`

`BookBlock.__delslice__()`

`x.__delslice__(i, j)` <==> `del x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

`BookBlock.__eq__()`

`x.__eq__(y)` <==> `x==y`

Inherited from `__builtin__.list`

`BookBlock.__ge__()`

`x.__ge__(y)` <==> `x>=y`

Inherited from `__builtin__.list`

`BookBlock.__getitem__()`

`x.__getitem__(y)` <==> `x[y]`

Inherited from `__builtin__.list`

`BookBlock.__getslice__()`

`x.__getslice__(i, j)` <==> `x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

`BookBlock.__gt__()`
`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.list`

`BookBlock.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Inherited from `__builtin__.list`

`BookBlock.__imul__()`
`x.__imul__(y) <==> x*=y`

Inherited from `__builtin__.list`

`BookBlock.__iter__()` <==> `iter(x)`
 Inherited from `__builtin__.list`

`BookBlock.__le__()`
`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.list`

`BookBlock.__len__()` <==> `len(x)`
 Inherited from `__builtin__.list`

`BookBlock.__lt__()`
`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.list`

`BookBlock.__mul__()`
`x.__mul__(n) <==> x*n`

Inherited from `__builtin__.list`

`BookBlock.__ne__()`
`x.__ne__(y) <==> x!=y`

Inherited from `__builtin__.list`

`BookBlock.__repr__()`
 Inherited from `lilypondfiletools.NonattributedBlock`

`BookBlock.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Inherited from `__builtin__.list`

`BookBlock.__rmul__()`
`x.__rmul__(n) <==> n*x`

Inherited from `__builtin__.list`

`BookBlock.__setitem__()`
`x.__setitem__(i, y) <==> x[i]=y`

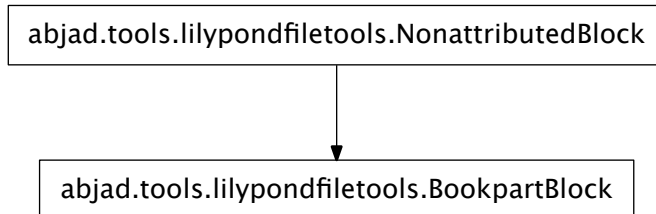
Inherited from `__builtin__.list`

`BookBlock.__setslice__()`
`x.__setslice__(i, j, y) <==> x[i:j]=y`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

`lilypondfiletools.BookpartBlock`



class `lilypondfiletools.BookpartBlock`

New in version 2.0. Abjad model of LilyPond input file bookpart block:

```

>>> bookpart_block = lilypondfiletools.BookpartBlock()

>>> bookpart_block
BookpartBlock()

>>> f(bookpart_block)
\bookpart {}
  
```

Return bookpart block.

Read-only Properties

`BookpartBlock.lilypond_format`

Inherited from `lilypondfiletools.NonattributedBlock`

Read/write Properties

`BookpartBlock.is_formatted_when_empty`

Inherited from `lilypondfiletools.NonattributedBlock`

Methods

`BookpartBlock.append()`

`L.append(object)` – append object to end

Inherited from `__builtin__.list`

`BookpartBlock.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`BookpartBlock.extend()`

`L.extend(iterable)` – extend list by appending elements from the iterable

Inherited from `__builtin__.list`

`BookpartBlock.index(value[, start[, stop]])` → integer – return first index of value.
 Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`BookpartBlock.insert()`
`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`BookpartBlock.pop([index])` → item – remove and return item at index (default last).
 Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`BookpartBlock.remove()`
`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`BookpartBlock.reverse()`
`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`BookpartBlock.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`BookpartBlock.__add__()`
`x.__add__(y) <==> x+y`

Inherited from `__builtin__.list`

`BookpartBlock.__contains__()`
`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.list`

`BookpartBlock.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Inherited from `__builtin__.list`

`BookpartBlock.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

`BookpartBlock.__eq__()`
`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.list`

`BookpartBlock.__ge__()`
`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.list`

`BookpartBlock.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.list`

`BookpartBlock.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

`BookpartBlock.__gt__()`
`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.list`

`BookpartBlock.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Inherited from `__builtin__.list`

`BookpartBlock.__imul__()`
`x.__imul__(y) <==> x*=y`

Inherited from `__builtin__.list`

`BookpartBlock.__iter__()` <==> `iter(x)`
 Inherited from `__builtin__.list`

`BookpartBlock.__le__()`
`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.list`

`BookpartBlock.__len__()` <==> `len(x)`
 Inherited from `__builtin__.list`

`BookpartBlock.__lt__()`
`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.list`

`BookpartBlock.__mul__()`
`x.__mul__(n) <==> x*n`

Inherited from `__builtin__.list`

`BookpartBlock.__ne__()`
`x.__ne__(y) <==> x!=y`

Inherited from `__builtin__.list`

`BookpartBlock.__repr__()`
 Inherited from `lilypondfiletools.NonattributedBlock`

`BookpartBlock.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Inherited from `__builtin__.list`

`BookpartBlock.__rmul__()`
`x.__rmul__(n) <==> n*x`

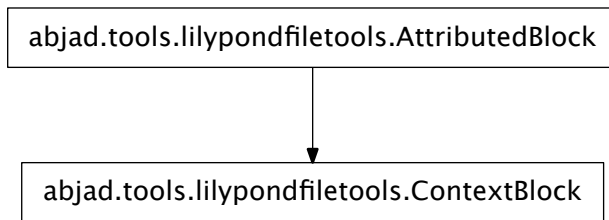
Inherited from `__builtin__.list`

`BookpartBlock.__setitem__()`
`x.__setitem__(i, y) <==> x[i]=y`

Inherited from `__builtin__.list`

BookpartBlock.__setslice__()
 x.__setslice__(i, j, y) <==> x[i:j]=y
 Use of negative indices is not supported.
 Inherited from __builtin__.list

lilypondfiletools.ContextBlock



```

class lilypondfiletools.ContextBlock(context_name=None)
  New in version 2.5. Abjad model of LilyPond input file context block:

  >>> context_block = lilypondfiletools.ContextBlock()

  >>> context_block
  ContextBlock()

  >>> context_block.context_name = 'Score'
  >>> context_block.override.bar_number.transparent = True
  >>> scheme = schemetools.Scheme('end-of-line-invisible')
  >>> context_block.override.time_signature.break_visibility = scheme
  >>> context_block.set.proportionalNotationDuration = schemetools.SchemeMoment((1, 45))

  >>> f(context_block)
  \context {
    \Score
    \override BarNumber #'transparent = ##t
    \override TimeSignature #'break-visibility = #end-of-line-invisible
    proportionalNotationDuration = #{ly:make-moment 1 45}
  }

  Return context block.
  
```

Read-only Properties

ContextBlock.**accepts**
 ContextBlock.**engraver_consists**
 ContextBlock.**engraver_removals**
 ContextBlock.**lilypond_format**
 Inherited from lilypondfiletools.AttributedBlock

`ContextBlock.override`

Read-only reference to LilyPond grob override component plug-in.

`ContextBlock.set`

Read-only reference LilyPond context setting component plug-in.

Read/write Properties

`ContextBlock.context_name`

Read / write context name.

`ContextBlock.is_formatted_when_empty`

Inherited from `lilypondfiletools.AttributedBlock`

`ContextBlock.name`

Read / write name.

`ContextBlock.type`

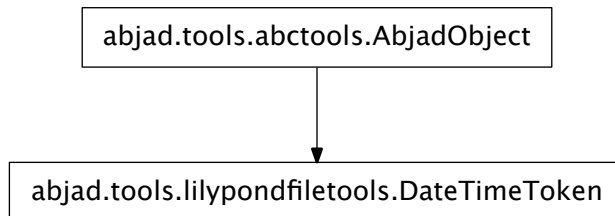
Read / write type.

Special Methods

`ContextBlock.__repr__()`

Inherited from `lilypondfiletools.AttributedBlock`

`lilypondfiletools.DateTimeToken`



class `lilypondfiletools.DateTimeToken`

New in version 2.0. Date time token:

```
>>> lilypondfiletools.DateTimeToken()
DateTimeToken(...)
```

Return date / time token.

Read-only Properties

`DateTimeToken.lilypond_format`

Format contribution of date time token:

```
>>> lilypondfiletools.DateTimeToken().lilypond_format
'...'
```

Return string.

`DateTimeToken.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`DateTimeToken.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DateTimeToken.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DateTimeToken.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DateTimeToken.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DateTimeToken.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DateTimeToken.__ne__(arg)`

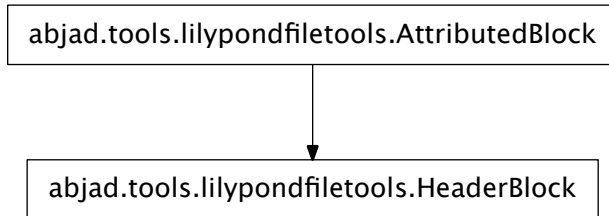
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DateTimeToken.__repr__()`

`lilypondfiletools.HeaderBlock`



`class lilypondfiletools.HeaderBlock`

New in version 2.0. Abjad model of LilyPond input file header block:

```

>>> header_block = lilypondfiletools.HeaderBlock()

>>> header_block
HeaderBlock()

>>> header_block.composer = markuptools.Markup('Josquin')
>>> header_block.title = markuptools.Markup('Missa sexti tonus')

>>> f(header_block)
\header {
  composer = \markup { Josquin }
  title = \markup { Missa sexti tonus }
}
  
```

Return header block.

Read-only Properties

`HeaderBlock.lilypond_format`

Inherited from `lilypondfiletools.AttributedBlock`

Read/write Properties

`HeaderBlock.is_formatted_when_empty`

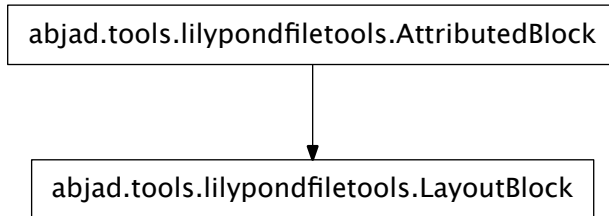
Inherited from `lilypondfiletools.AttributedBlock`

Special Methods

`HeaderBlock.__repr__()`

Inherited from `lilypondfiletools.AttributedBlock`

lilypondfiletools.LayoutBlock



class lilypondfiletools.LayoutBlock

New in version 2.0. Abjad model of LilyPond input file layout block:

```

>>> layout_block = lilypondfiletools.LayoutBlock()

>>> layout_block
LayoutBlock()

>>> layout_block.indent = 0
>>> layout_block.ragged_right = True

>>> f(layout_block)
\layout {
  indent = #0
  ragged-right = ##t
}
  
```

Return layout block.

Read-only Properties

LayoutBlock.context_blocks

Read-only list of context blocks:

```

>>> layout_block = lilypondfiletools.LayoutBlock()

>>> context_block = lilypondfiletools.ContextBlock('Score')
>>> context_block.override.bar_number.transparent = True

>>> scheme_expr = schemetools.Scheme('end-of-line-invisible')
>>> context_block.override.time_signature.break_visibility = scheme_expr
>>> layout_block.context_blocks.append(context_block)

>>> f(layout_block)
\layout {
  \context {
    \Score
    \override BarNumber #'transparent = ##t
    \override TimeSignature #'break-visibility = #end-of-line-invisible
  }
}
  
```

Return list.

`LayoutBlock.contexts`

DEPRECATED. USE `CONTEXT_BLOCKS` INSTEAD.

`LayoutBlock.lilypond_format`

Inherited from `lilypondfiletools.AttributedBlock`

Read/write Properties

`LayoutBlock.is_formatted_when_empty`

Inherited from `lilypondfiletools.AttributedBlock`

Special Methods

`LayoutBlock.__repr__()`

Inherited from `lilypondfiletools.AttributedBlock`

`lilypondfiletools.LilyPondFile`

`abjad.tools.lilypondfiletools.LilyPondFile`

`class lilypondfiletools.LilyPondFile`

New in version 2.0. Abjad model of LilyPond input file:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> lilypond_file = lilypondfiletools.make_basic_lilypond_file(staff)
>>> lilypond_file.file_initial_user_comments.append('File construct as an example.')
>>> lilypond_file.file_initial_user_comments.append('Parts shown here for positioning.')
>>> lilypond_file.file_initial_user_includes.append('external-settings-file-1.ly')
>>> lilypond_file.file_initial_user_includes.append('external-settings-file-2.ly')
>>> lilypond_file.default_paper_size = 'letter', 'portrait'
>>> lilypond_file.global_staff_size = 16
>>> lilypond_file.header_block.composer = markuptools.Markup('Josquin')
>>> lilypond_file.header_block.title = markuptools.Markup('Missa sexti tonus')
>>> lilypond_file.layout_block.indent = 0
>>> lilypond_file.layout_block.left_margin = 15
>>> lilypond_file.paper_block.oddFooterMarkup = markuptools.Markup('The odd-page footer')
>>> lilypond_file.paper_block.evenFooterMarkup = markuptools.Markup('The even-page footer')

>>> f(lilypond_file)
% Abjad revision 3719
% 2010-09-24 09:01

% File construct as an example.
% Parts shown here for positioning.

\version "2.13.32"
\include "english.ly"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"
```

```
\include "external-settings-file-1.ly"
\include "external-settings-file-2.ly"

#(set-default-paper-size "letter" 'portrait)
#(set-global-staff-size 16)

\header {
  composer = \markup { Josquin }
  title = \markup { Missa sexti tonus }
}

\layout {
  indent = #0
  left-margin = #15
}

\paper {
  evenFooterMarkup = \markup { The even-page footer }
  oddFooterMarkup = \markup { The odd-page footer }
}

\new Staff {
  c'8
  d'8
  e'8
  f'8
}
```

Read-only Properties

`LilyPondFile.lilypond_format`

Format-time contribution of LilyPond file.

Read/write Properties

`LilyPondFile.default_paper_size`

LilyPond default paper size.

`LilyPondFile.file_initial_system_comments`

Read-only list of file-initial system comments.

`LilyPondFile.file_initial_system_includes`

List of file-initial system include commands.

`LilyPondFile.file_initial_user_comments`

Read-only list of file-initial user comments.

`LilyPondFile.file_initial_user_includes`

List of file-initial user include commands.

`LilyPondFile.global_staff_size`

LilyPond global staff size.

Methods

`LilyPondFile.append()`

`L.append(object)` – append object to end

Inherited from `__builtin__.list`

`LilyPondFile.count(value)` → integer – return number of occurrences of value
 Inherited from `__builtin__.list`

`LilyPondFile.extend()`
`L.extend(iterable)` – extend list by appending elements from the iterable
 Inherited from `__builtin__.list`

`LilyPondFile.index(value[, start[, stop]])` → integer – return first index of value.
 Raises `ValueError` if the value is not present.
 Inherited from `__builtin__.list`

`LilyPondFile.insert()`
`L.insert(index, object)` – insert object before index
 Inherited from `__builtin__.list`

`LilyPondFile.pop([index])` → item – remove and return item at index (default last).
 Raises `IndexError` if list is empty or index is out of range.
 Inherited from `__builtin__.list`

`LilyPondFile.remove()`
`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.
 Inherited from `__builtin__.list`

`LilyPondFile.reverse()`
`L.reverse()` – reverse *IN PLACE*
 Inherited from `__builtin__.list`

`LilyPondFile.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`
 Inherited from `__builtin__.list`

Special Methods

`LilyPondFile.__add__()`
`x.__add__(y) <==> x+y`
 Inherited from `__builtin__.list`

`LilyPondFile.__contains__()`
`x.__contains__(y) <==> y in x`
 Inherited from `__builtin__.list`

`LilyPondFile.__delitem__()`
`x.__delitem__(y) <==> del x[y]`
 Inherited from `__builtin__.list`

`LilyPondFile.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

`LilyPondFile.__eq__()`
`x.__eq__(y) <==> x==y`
 Inherited from `__builtin__.list`

```

LilyPondFile.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.list

LilyPondFile.__getitem__()
    x.__getitem__(y) <==> x[y]
    Inherited from __builtin__.list

LilyPondFile.__getslice__()
    x.__getslice__(i, j) <==> x[i:j]
    Use of negative indices is not supported.
    Inherited from __builtin__.list

LilyPondFile.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.list

LilyPondFile.__iadd__()
    x.__iadd__(y) <==> x+=y
    Inherited from __builtin__.list

LilyPondFile.__imul__()
    x.__imul__(y) <==> x*=y
    Inherited from __builtin__.list

LilyPondFile.__iter__() <==> iter(x)
    Inherited from __builtin__.list

LilyPondFile.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.list

LilyPondFile.__len__() <==> len(x)
    Inherited from __builtin__.list

LilyPondFile.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.list

LilyPondFile.__mul__()
    x.__mul__(n) <==> x*n
    Inherited from __builtin__.list

LilyPondFile.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.list

LilyPondFile.__repr__()

LilyPondFile.__reversed__()
    L.__reversed__() – return a reverse iterator over the list
    Inherited from __builtin__.list

```

```
LilyPondFile.__rmul__ ()
    x.__rmul__(n) <==> n*x

    Inherited from __builtin__.list

LilyPondFile.__setitem__ ()
    x.__setitem__(i, y) <==> x[i]=y

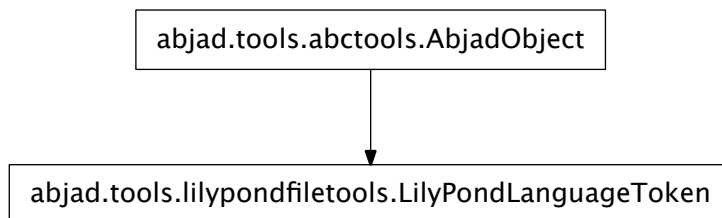
    Inherited from __builtin__.list

LilyPondFile.__setslice__ ()
    x.__setslice__(i, j, y) <==> x[i:j]=y

    Use of negative indices is not supported.

    Inherited from __builtin__.list
```

lilypondfiletools.LilyPondLanguageToken



class lilypondfiletools.LilyPondLanguageToken

New in version 2.0. LilyPond language token:

```
>>> lilypondfiletools.LilyPondLanguageToken ()
LilyPondLanguageToken('english')
```

Return LilyPond language token. Changed in version 2.9: format with LilyPond `\language` command instead of LilyPond `\include` command.

Read-only Properties

LilyPondLanguageToken.lilypond_format

Format contribution of LilyPond language token:

```
>>> lilypondfiletools.LilyPondLanguageToken().lilypond_format
'\language "english"'
```

Return string.

LilyPondLanguageToken.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`LilyPondLanguageToken.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`LilyPondLanguageToken.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`LilyPondLanguageToken.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

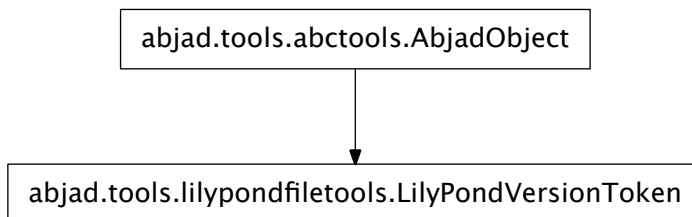
`LilyPondLanguageToken.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`LilyPondLanguageToken.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`LilyPondLanguageToken.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`LilyPondLanguageToken.__repr__()`

`lilypondfiletools.LilyPondVersionToken`



class `lilypondfiletools.LilyPondVersionToken`

New in version 2.0. LilyPond version token:

```
>>> lilypondfiletools.LilyPondVersionToken()  
LilyPondVersionToken(\version "...")
```

Return LilyPond version token.

Read-only Properties

`LilyPondVersionToken.lilypond_format`

Format contribution of LilyPond version token:

```
>>> lilypondfiletools.LilyPondVersionToken().lilypond_format  
'\\version "..."
```

Return string.

`LilyPondVersionToken.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`LilyPondVersionToken.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`LilyPondVersionToken.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondVersionToken.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`LilyPondVersionToken.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondVersionToken.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondVersionToken.__ne__(arg)`

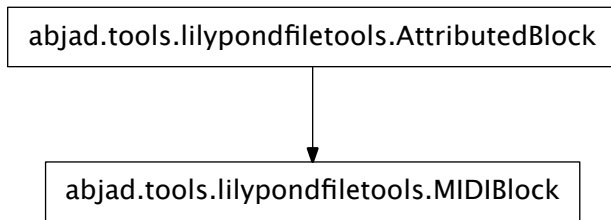
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`LilyPondVersionToken.__repr__()`

`lilypondfiletools.MIDIBlock`



`class lilypondfiletools.MIDIBlock`

New in version 2.0. Abjad model of LilyPond input file MIDI block:

```

>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> score = Score([staff])
>>> lilypond_file = lilypondfiletools.make_basic_lilypond_file(score)

>>> lilypond_file.score_block.append(lilypondfiletools.MIDIBlock())

>>> layout_block = lilypondfiletools.LayoutBlock()
>>> layout_block.is_formatted_when_empty = True
>>> lilypond_file.score_block.append(layout_block)

>>> f(lilypond_file.score_block)
\score {
  \new Score <<
    \new Staff {
      c'4
      d'4
      e'4
      f'4
    }
  >>
  \midi {}
  \layout {}
}
  
```

MIDI blocks are formatted even when they are empty.

The example here appends MIDI and layout blocks to a score block. Doing this allows LilyPond to create both MIDI and PDF output from a single input file.

Read the LilyPond docs on LilyPond file structure for the details as to why this is the case.

Read-only Properties

`MIDIBlock.lilypond_format`

Inherited from `lilypondfiletools.AttributedBlock`

Read/write Properties

`MIDIBlock.is_formatted_when_empty`

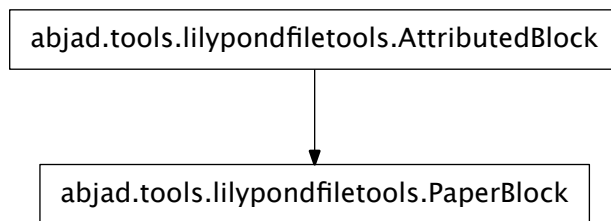
Inherited from `lilypondfiletools.AttributedBlock`

Special Methods

`MIDIBlock.__repr__()`

Inherited from `lilypondfiletools.AttributedBlock`

`lilypondfiletools.PaperBlock`



class `lilypondfiletools.PaperBlock`

New in version 2.0. Abjad model of LilyPond input file paper block:

```
>>> paper_block = lilypondfiletools.PaperBlock()
```

```
>>> paper_block
PaperBlock()
```

```
>>> paper_block.print_page_number = True
>>> paper_block.print_first_page_number = False
```

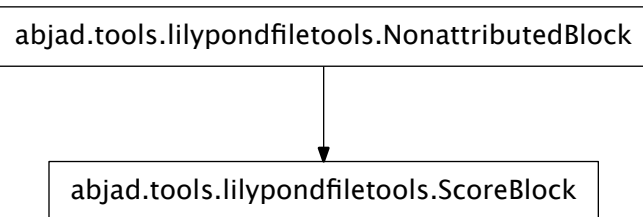
```
>>> f(paper_block)
\paper {
  print-first-page-number = ##f
  print-page-number = ##t
}
```

Return paper block.

Read-only Properties

`PaperBlock.lilypond_format`

Inherited from `lilypondfiletools.AttributedBlock`

Read/write Properties`PaperBlock.is_formatted_when_empty`Inherited from `lilypondfiletools.AttributedBlock``PaperBlock.minimal_page_breaking`**Special Methods**`PaperBlock.__repr__()`Inherited from `lilypondfiletools.AttributedBlock`**`lilypondfiletools.ScoreBlock`****class `lilypondfiletools.ScoreBlock`**

New in version 2.0. Abjad model of LilyPond input file score block:

```
>>> score_block = lilypondfiletools.ScoreBlock()
```

```
>>> score_block
ScoreBlock()
```

```

>>> score_block.append(Staff([]))
>>> f(score_block)
\score {
  \new Staff {
  }
}

```

ScoreBlocks does not format when empty, as this generates a parser error in LilyPond:

```

>>> score_block = lilypondfiletools.ScoreBlock()
>>> score_block.lilypond_format == ''
True

```

Return score block.

Read-only Properties`ScoreBlock.lilypond_format`Inherited from `lilypondfiletools.NonattributedBlock`

Read/write Properties

`ScoreBlock.is_formatted_when_empty`

Inherited from `lilypondfiletools.NonattributedBlock`

Methods

`ScoreBlock.append()`

`L.append(object)` – append object to end

Inherited from `__builtin__.list`

`ScoreBlock.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`ScoreBlock.extend()`

`L.extend(iterable)` – extend list by appending elements from the iterable

Inherited from `__builtin__.list`

`ScoreBlock.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`ScoreBlock.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`ScoreBlock.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`ScoreBlock.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`ScoreBlock.reverse()`

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`ScoreBlock.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`ScoreBlock.__add__()`

`x.__add__(y)` <==> `x+y`

Inherited from `__builtin__.list`

`ScoreBlock.__contains__()`

`x.__contains__(y)` <==> `y in x`

Inherited from `__builtin__.list`

`ScoreBlock.__delitem__()`

`x.__delitem__(y)` <==> `del x[y]`

Inherited from `__builtin__.list`

`ScoreBlock.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

`ScoreBlock.__eq__()`
`x.__eq__(y) <==> x==y`
 Inherited from `__builtin__.list`

`ScoreBlock.__ge__()`
`x.__ge__(y) <==> x>=y`
 Inherited from `__builtin__.list`

`ScoreBlock.__getitem__()`
`x.__getitem__(y) <==> x[y]`
 Inherited from `__builtin__.list`

`ScoreBlock.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

`ScoreBlock.__gt__()`
`x.__gt__(y) <==> x>y`
 Inherited from `__builtin__.list`

`ScoreBlock.__iadd__()`
`x.__iadd__(y) <==> x+=y`
 Inherited from `__builtin__.list`

`ScoreBlock.__imul__()`
`x.__imul__(y) <==> x*=y`
 Inherited from `__builtin__.list`

`ScoreBlock.__iter__()` <==> `iter(x)`
 Inherited from `__builtin__.list`

`ScoreBlock.__le__()`
`x.__le__(y) <==> x<=y`
 Inherited from `__builtin__.list`

`ScoreBlock.__len__()` <==> `len(x)`
 Inherited from `__builtin__.list`

`ScoreBlock.__lt__()`
`x.__lt__(y) <==> x<y`
 Inherited from `__builtin__.list`

`ScoreBlock.__mul__()`
`x.__mul__(n) <==> x*n`
 Inherited from `__builtin__.list`

```
ScoreBlock.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.list

ScoreBlock.__repr__()
    Inherited from lilypondfiletools.NonattributedBlock

ScoreBlock.__reversed__()
    L.__reversed__() – return a reverse iterator over the list
    Inherited from __builtin__.list

ScoreBlock.__rmul__()
    x.__rmul__(n) <==> n*x
    Inherited from __builtin__.list

ScoreBlock.__setitem__()
    x.__setitem__(i, y) <==> x[i]=y
    Inherited from __builtin__.list

ScoreBlock.__setslice__()
    x.__setslice__(i, j, y) <==> x[i:j]=y
    Use of negative indices is not supported.
    Inherited from __builtin__.list
```

functions

`lilypondfiletools.make_basic_lilypond_file`

`lilypondfiletools.make_basic_lilypond_file` (*music=None*)

New in version 2.0. Make basic LilyPond file with *music*:

```
>>> score = Score([Staff("c'8 d'8 e'8 f'8")])
>>> lilypond_file = lilypondfiletools.make_basic_lilypond_file(score)
>>> lilypond_file.header_block.composer = markuptools.Markup('Josquin')
>>> lilypond_file.layout_block.indent = 0
>>> lilypond_file.paper_block.top_margin = 15
>>> lilypond_file.paper_block.left_margin = 15

>>> f(lilypond_file)
\header {
  composer = \markup { Josquin }
}

\layout {
  indent = #0
}

\paper {
  left-margin = #15
  top-margin = #15
}

\new Score <<
  \new Staff {
```

```

        c' 8
        d' 8
        e' 8
        f' 8
    }
>>

```

Equip LilyPond file with header, layout and paper blocks.

Return LilyPond file.

`lilypondfiletools.make_floating_time_signature_lilypond_file`

`lilypondfiletools.make_floating_time_signature_lilypond_file` (*music=None*)

New in version 2.10. Make floating time signature LilyPond file from *music*.

Function creates a basic LilyPond file.

Function then applies many layout settings.

[View source here](#) for the complete inventory of settings applied.

Returns LilyPond file object.

`lilypondfiletools.make_time_signature_context_block`

`lilypondfiletools.make_time_signature_context_block` (*font_size=3, padding=4*)

New in version 2.9. Make time signature context block:

```

>>> context_block = lilypondfiletools.make_time_signature_context_block()

>>> f(context_block)
\context {
  \type Engraver_group
  \name TimeSignatureContext
  \consists Axis_group_engraver
  \consists Time_signature_engraver
  \override TimeSignature #'X-extent = #'(0 . 0)
  \override TimeSignature #'X-offset = #ly:self-alignment-interface::x-aligned-on-self
  \override TimeSignature #'Y-extent = #'(0 . 0)
  \override TimeSignature #'break-align-symbol = ##f
  \override TimeSignature #'break-visibility = #end-of-line-invisible
  \override TimeSignature #'font-size = #3
  \override TimeSignature #'self-alignment-X = #center
  \override VerticalAxisGroup #'default-staff-staff-spacing = #'(
    (basic_distance . 0) (minimum_distance . 0) (padding . 4) (stretchability . 0))
}

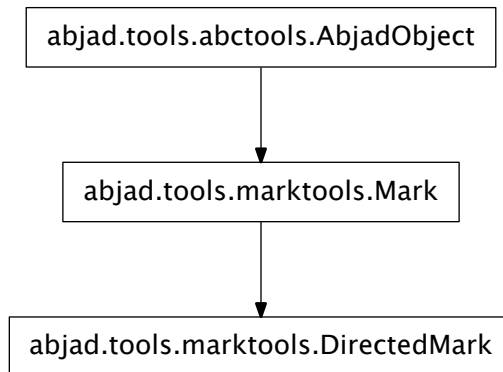
```

Return context block.

marktools

abstract classes

marktools.DirectedMark



class marktools.**DirectedMark** (*args, **kwargs)

Abstract base class of Marks which possess a vertical, typographic direction, i.e. above or below the staff.

Read-only Properties

DirectedMark.**start_component**

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)
```

```
>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from marktools.Mark

DirectedMark.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from abctools.AbjadObject

Read/write Properties

DirectedMark.**direction**

Methods

DirectedMark.**attach**(*start_component*)

Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()
```

```
>>> mark.attach(note)
Mark() (c'4)
```

```
>>> mark.start_component
Note("c'4")
```

Return mark.

Inherited from marktools.Mark

DirectedMark.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)
```

```
>>> mark.start_component
Note("c'4")
```

```
>>> mark.detach()
Mark()
```

```
>>> mark.start_component is None
True
```

Return mark.

Inherited from marktools.Mark

Special Methods

DirectedMark.**__call__**(*args)

Inherited from marktools.Mark

DirectedMark.**__delattr__**(*args)

Inherited from marktools.Mark

DirectedMark.**__eq__**(arg)

Inherited from marktools.Mark

DirectedMark.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

DirectedMark.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

DirectedMark.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DirectedMark.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DirectedMark.__ne__(arg)`

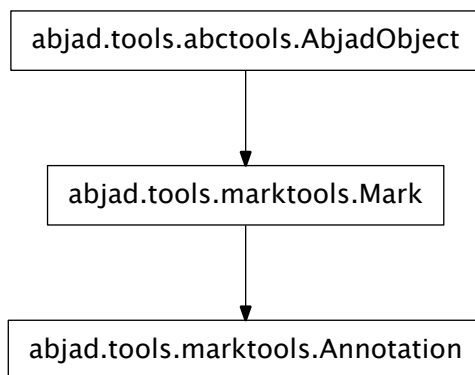
Inherited from `marktools.Mark`

`DirectedMark.__repr__()`

Inherited from `marktools.Mark`

concrete classes

`marktools.Annotation`



class `marktools.Annotation(*args)`

New in version 2.0. User-defined annotation:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> f(staff)
\new Staff {
  c'8
  d'8
  e'8
  f'8
}

>>> pitch = pitchtools.NamedChromaticPitch('ds')
>>> marktools.Annotation('special pitch', pitch)(staff[0])
Annotation('special pitch', NamedChromaticPitch('ds'))(c'8)
  
```

```
>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
```

Annotations contribute no formatting.

Annotations implement `__slots__`.

Read-only Properties

Annotation.**start_component**

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Annotation.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Annotation.**name**

Get name of annotation:

```
>>> pitch = pitchtools.NamedChromaticPitch('ds')
>>> annotation = marktools.Annotation('special_pitch', pitch)
>>> annotation.name
'special_pitch'
```

Set name of annotation:

```
>>> annotation.name = 'revised special pitch'
>>> annotation.name
'revised special pitch'
```

Set string.

Annotation.**value**

Get value of annotation:

```
>>> annotation.value
NamedChromaticPitch('ds')
```

Set value of annotation:

```
>>> annotation.value = pitchtools.NamedChromaticPitch('e')
>>> annotation.value
NamedChromaticPitch('e')
```

Set arbitrary object.

Methods

Annotation.**attach**(*start_component*)

Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()

>>> mark.attach(note)
Mark() (c'4)

>>> mark.start_component
Note("c'4")
```

Return mark.

Inherited from `marktools.Mark`

Annotation.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")

>>> mark.detach()
Mark()

>>> mark.start_component is None
True
```

Return mark.

Inherited from `marktools.Mark`

Special Methods

Annotation.**__call__**(*args)

Inherited from `marktools.Mark`

Annotation.**__delattr__**(*args)

Inherited from `marktools.Mark`

Annotation.**__eq__**(arg)

Annotation.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Annotation.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

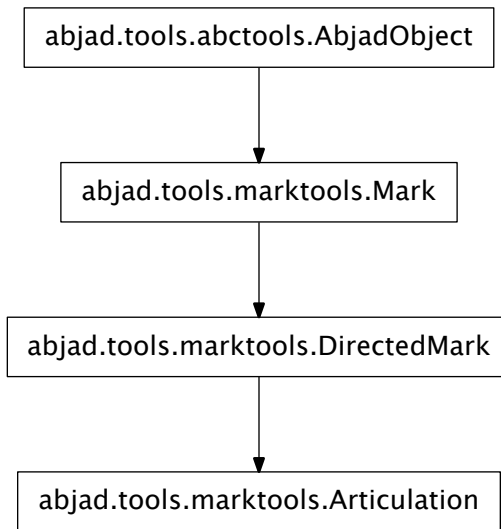
`Annotation.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Annotation.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Annotation.__ne__(arg)`
 Inherited from `marktools.Mark`

`Annotation.__repr__()`
 Inherited from `marktools.Mark`

marktools.Articulation



```

class marktools.Articulation(*args)
  Abjad model of musical articulation:

  >>> note = Note("c'4")

  >>> marktools.Articulation('staccato')(note)
  Articulation('staccato')(c'4)

  >>> f(note)
  c'4 -\staccato

  Articulations implement __slots__.
  
```

Read-only Properties

Articulation.lilypond_format

Read-only LilyPond format string of articulation:

```
>>> articulation = marktools.Articulation('marcato', Up)
>>> articulation.lilypond_format
'^\marcato'
```

Return string.

Articulation.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Articulation.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Articulation.direction

Inherited from `marktools.DirectedMark`

Articulation.name

Get name of articulation:

```
>>> articulation = marktools.Articulation('staccato', Up)
>>> articulation.name
'staccato'
```

Set name of articulation:

```
>>> articulation.name = 'marcato'
>>> articulation.name
'marcato'
```

Set string.

Methods

Articulation.attach(*start_component*)

Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()

>>> mark.attach(note)
Mark()(c'4)

>>> mark.start_component
Note("c'4")
```

Return mark.

Inherited from `marktools.Mark`

`Articulation.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")

>>> mark.detach()
Mark()

>>> mark.start_component is None
True
```

Return mark.

Inherited from `marktools.Mark`

Special Methods

`Articulation.__call__(*args)`

Inherited from `marktools.Mark`

`Articulation.__delattr__(*args)`

Inherited from `marktools.Mark`

`Articulation.__eq__(expr)`

`Articulation.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Articulation.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Articulation.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Articulation.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Articulation.__ne__(arg)`

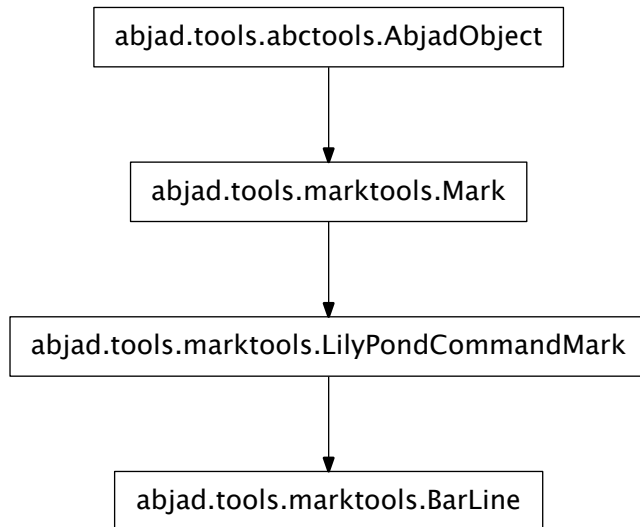
Inherited from `marktools.Mark`

`Articulation.__repr__()`

Inherited from `marktools.Mark`

Articulation.__str__()

marktools.BarLine



class marktools.**BarLine** (*bar_line_string='|', format_slot='after'*)

New in version 2.4. Abjad model of bar line:

```

>>> staff = Staff("c'4 d'4 e'4 f'4")

>>> bar_line = marktools.BarLine('||.')(staff[-1])

>>> bar_line
BarLine('||.')(f'4)

>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
  \bar "||."
}
  
```

Return bar line.

Read-only Properties

BarLine.lilypond_format

Read-only LilyPond input format of LilyPond command mark:


```
>>> note = Note("c'4")
>>> lilypond_command = marktools.LilyPondCommandMark('slurDotted')(note)
>>> lilypond_command.lilypond_format
'\slurDotted'
```

Return string.

Inherited from `marktools.LilyPondCommandMark`

`BarLine.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)
```

```
>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`BarLine.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`BarLine.bar_line_string`

Get bar line string of bar line:

```
>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> bar_line = marktools.BarLine()(staff[-1])
>>> bar_line.bar_line_string
'|'
```

Set bar line string of bar line:

```
>>> bar_line.bar_line_string = '|.'
>>> bar_line.bar_line_string
'|.'
```

```
>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
  \bar "|."
}
```

Set string.

`BarLine.command_name`

Get command name of LilyPond command mark:

```
>>> lilypond_command = marktools.LilyPondCommandMark('slurDotted')
>>> lilypond_command.command_name
'slurDotted'
```

Set command name of LilyPond command mark:

```
>>> lilypond_command.command_name = 'slurDashed'
>>> lilypond_command.command_name
'slurDashed'
```

Set string.

Inherited from `marktools.LilyPondCommandMark`

BarLine.format_slot

New in version 2.3. Get format slot of LilyPond command mark:

```
>>> note = Note("c'4")
>>> lilypond_command = marktools.LilyPondCommandMark('break', 'after')
>>> lilypond_command.format_slot
'after'
```

Set format slot of LilyPond command mark:

```
>>> note = Note("c'4")
>>> lilypond_command = marktools.LilyPondCommandMark('break', 'after')
>>> lilypond_command.format_slot = 'before'
>>> lilypond_command.format_slot
'before'
```

Set to 'before', 'after', 'opening', 'closing', 'right' or none.

Inherited from `marktools.LilyPondCommandMark`

Methods

BarLine.attach(*start_component*)

Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()

>>> mark.attach(note)
Mark() (c'4)

>>> mark.start_component
Note("c'4")
```

Return mark.

Inherited from `marktools.Mark`

BarLine.detach()

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")

>>> mark.detach()
Mark()

>>> mark.start_component is None
True
```

Return mark.

Inherited from `marktools.Mark`

Special Methods

`BarLine.__call__(*args)`

Inherited from `marktools.Mark`

`BarLine.__delattr__(*args)`

Inherited from `marktools.Mark`

`BarLine.__eq__(arg)`

Inherited from `marktools.LilyPondCommandMark`

`BarLine.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BarLine.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BarLine.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BarLine.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

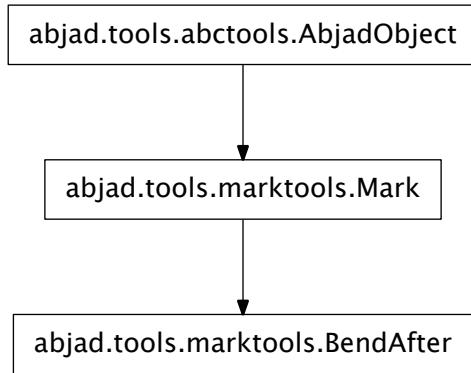
`BarLine.__ne__(arg)`

Inherited from `marktools.Mark`

`BarLine.__repr__()`

Inherited from `marktools.Mark`

marktools.BendAfter



```

class marktools.BendAfter (*args)
    Abjad model of a fall or doit:

    >>> note = Note("c'4")

    >>> marktools.BendAfter(-4)(note)
    BendAfter(-4.0)(c'4)

    >>> f(note)
    c'4 - \bendAfter #'-4.0

    BendAfter implements __slots__.
  
```

Read-only Properties

BendAfter.lilypond_format

Read-only LilyPond format string:

```

>>> bend = marktools.BendAfter(-4)
>>> bend.lilypond_format
"- \bendAfter #'-4.0"
  
```

Return string.

BendAfter.start_component

Read-only reference to mark start component:

```

>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
  
```

Return component or none.

Inherited from `marktools.Mark`

`BendAfter.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`BendAfter.bend_amount`

Get bend amount:

```
>>> bend = marktools.BendAfter(8)
>>> bend.bend_amount
8.0
```

Set bend amount:

```
>>> bend.bend_amount = -4
>>> bend.bend_amount
-4.0
```

Set float.

Methods

`BendAfter.attach(start_component)`

Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()

>>> mark.attach(note)
Mark() (c'4)

>>> mark.start_component
Note("c'4")
```

Return mark.

Inherited from `marktools.Mark`

`BendAfter.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")

>>> mark.detach()
Mark()

>>> mark.start_component is None
True
```

Return mark.

Inherited from `marktools.Mark`

Special Methods

`BendAfter.__call__(*args)`

Inherited from `marktools.Mark`

`BendAfter.__delattr__(*args)`

Inherited from `marktools.Mark`

`BendAfter.__eq__(expr)`

`BendAfter.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BendAfter.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BendAfter.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BendAfter.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

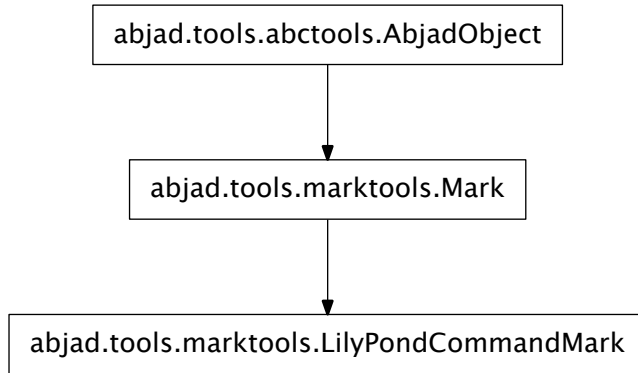
`BendAfter.__ne__(arg)`

Inherited from `marktools.Mark`

`BendAfter.__repr__()`

Inherited from `marktools.Mark`

`BendAfter.__str__()`

marktools.LilyPondCommandMark

class marktools.**LilyPondCommandMark** (*args)

New in version 2.0. LilyPond command mark:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff.leaves)

>>> lilypond_command = marktools.LilyPondCommandMark('slurDotted')(staff[0])

>>> f(staff)
\new Staff {
  \slurDotted
  c'8 (
  d'8
  e'8
  f'8 )
}

```

Initialize LilyPond command marks from command name; or from command name with format slot; or from another LilyPond command mark; or from another LilyPond command mark with format slot.

LilyPond command marks implement `__slots__`.

Read-only Properties

`LilyPondCommandMark.lilypond_format`

Read-only LilyPond input format of LilyPond command mark:

```

>>> note = Note("c'4")
>>> lilypond_command = marktools.LilyPondCommandMark('slurDotted')(note)
>>> lilypond_command.lilypond_format
'\slurDotted'

```

Return string.

`LilyPondCommandMark.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`LilyPondCommandMark.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`LilyPondCommandMark.command_name`
Get command name of LilyPond command mark:

```
>>> lilypond_command = marktools.LilyPondCommandMark('slurDotted')
>>> lilypond_command.command_name
'slurDotted'
```

Set command name of LilyPond command mark:

```
>>> lilypond_command.command_name = 'slurDashed'
>>> lilypond_command.command_name
'slurDashed'
```

Set string.

`LilyPondCommandMark.format_slot`
New in version 2.3. Get format slot of LilyPond command mark:

```
>>> note = Note("c'4")
>>> lilypond_command = marktools.LilyPondCommandMark('break', 'after')
>>> lilypond_command.format_slot
'after'
```

Set format slot of LilyPond command mark:

```
>>> note = Note("c'4")
>>> lilypond_command = marktools.LilyPondCommandMark('break', 'after')
>>> lilypond_command.format_slot = 'before'
>>> lilypond_command.format_slot
'before'
```

Set to 'before', 'after', 'opening', 'closing', 'right' or none.

Methods

`LilyPondCommandMark.attach(start_component)`
Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()

>>> mark.attach(note)
Mark()(c'4)
```



```
>>> mark.start_component
Note("c'4")
```

Return mark.

Inherited from `marktools.Mark`

`LilyPondCommandMark.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)
```

```
>>> mark.start_component
Note("c'4")
```

```
>>> mark.detach()
Mark()
```

```
>>> mark.start_component is None
True
```

Return mark.

Inherited from `marktools.Mark`

Special Methods

`LilyPondCommandMark.__call__(*args)`

Inherited from `marktools.Mark`

`LilyPondCommandMark.__delattr__(*args)`

Inherited from `marktools.Mark`

`LilyPondCommandMark.__eq__(arg)`

`LilyPondCommandMark.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondCommandMark.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`LilyPondCommandMark.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondCommandMark.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

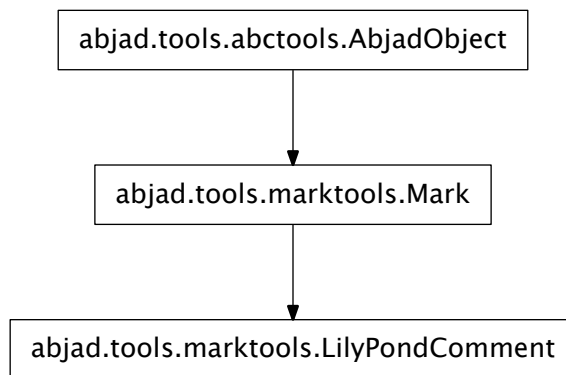
Inherited from `abctools.AbjadObject`

`LilyPondCommandMark.__ne__(arg)`

Inherited from `marktools.Mark`

`LilyPondCommandMark.__repr__()`
 Inherited from `marktools.Mark`

`marktools.LilyPondComment`



class `marktools.LilyPondComment` (*args)

New in version 2.0.Changed in version 2.3: Changed `Comment` to `LilyPondComment`. User-defined comment:

```

>>> note = Note("c'4")

>>> marktools.LilyPondComment('this is a comment')(note)
LilyPondComment('this is a comment')(c'4)

>>> f(note)
% this is a comment
c'4
    
```

Initialize `LilyPond` comment from contents string; or contents string and format slot; or from other `LilyPond` comment; or from other `LilyPond` comment and format slot.

`LilyPond` comments implement `__slots__`.

Read-only Properties

`LilyPondComment.lilypond_format`

Read-only `LilyPond` input format of comment:

```

>>> comment = marktools.LilyPondComment('this is a comment.')
>>> comment.lilypond_format
'% this is a comment.'
    
```

Return string.

`LilyPondComment.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

`LilyPondComment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`LilyPondComment.contents_string`

Get contents string of comment:

```
>>> comment = marktools.LilyPondComment('comment contents string')
>>> comment.contents_string
'comment contents string'
```

Set contents string of comment:

```
>>> comment.contents_string = 'new comment contents string'
>>> comment.contents_string
'new comment contents string'
```

Set string.

`LilyPondComment.format_slot`

New in version 2.3. Get format slot of LilyPond comment:

```
>>> note = Note("c'4")
>>> lilypond_comment = marktools.LilyPondComment('comment')
>>> lilypond_comment.format_slot
'before'
```

Set format slot of LilyPond comment:

```
>>> note = Note("c'4")
>>> lilypond_comment = marktools.LilyPondComment('comment')
>>> lilypond_comment.format_slot = 'after'
>>> lilypond_comment.format_slot
'after'
```

Set to 'before', 'after', 'opening', 'closing', 'right' or none.

Methods

`LilyPondComment.attach(start_component)`

Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()

>>> mark.attach(note)
Mark()(c'4)
```

```
>>> mark.start_component
Note("c'4")
```

Return mark.

Inherited from `marktools.Mark`

`LilyPondComment.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)
```

```
>>> mark.start_component
Note("c'4")
```

```
>>> mark.detach()
Mark()
```

```
>>> mark.start_component is None
True
```

Return mark.

Inherited from `marktools.Mark`

Special Methods

`LilyPondComment.__call__(*args)`

Inherited from `marktools.Mark`

`LilyPondComment.__delattr__(*args)`

Inherited from `marktools.Mark`

`LilyPondComment.__eq__(arg)`

`LilyPondComment.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondComment.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`LilyPondComment.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondComment.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

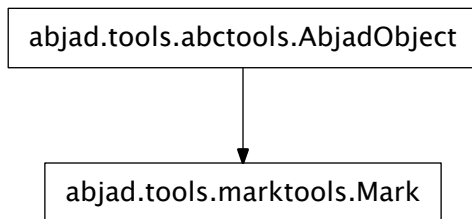
Inherited from `abctools.AbjadObject`

`LilyPondComment.__ne__(arg)`

Inherited from `marktools.Mark`

`LilyPondComment.__repr__()`
 Inherited from `marktools.Mark`

marktools.Mark



class `marktools.Mark(*args)`
 New in version 2.0. Abstract class from which concrete marks inherit:

```
>>> note = Note("c'4")

>>> marktools.Mark()(note)
Mark()(c'4)
```

Marks override `__call__` to attach to a note, rest or chord.

Marks are immutable.

Read-only Properties

`Mark.start_component`

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

`Mark.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`Mark.attach(start_component)`

Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()
```

```
>>> mark.attach(note)
Mark() (c'4)
```

```
>>> mark.start_component
Note("c'4")
```

Return mark.

Mark.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)
```

```
>>> mark.start_component
Note("c'4")
```

```
>>> mark.detach()
Mark()
```

```
>>> mark.start_component is None
True
```

Return mark.

Special Methods

Mark.**__call__**(*args)

Mark.**__delattr__**(*args)

Mark.**__eq__**(arg)

Mark.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Mark.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Mark.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Mark.**__lt__**(arg)

Abjad objects by default do not implement this method.

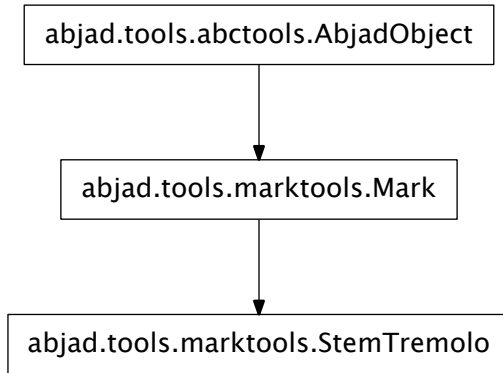
Raise exception.

Inherited from `abctools.AbjadObject`

Mark.**__ne__**(arg)

Mark.**__repr__**()

marktools.StemTremolo



class marktools.**StemTremolo** (*args)
 New in version 2.0. Abjad model of stem tremolo:

```

>>> note = Note("c'4")

>>> marktools.StemTremolo(16)(note)
StemTremolo(16)(c'4)

>>> f(note)
c'4 :16
  
```

Stem tremolos implement `__slots__`.

Read-only Properties

StemTremolo.lilypond_format

Read-only LilyPond format string:

```

>>> stem_tremolo = marktools.StemTremolo(16)
>>> stem_tremolo.lilypond_format
':16'
  
```

Return string.

StemTremolo.start_component

Read-only reference to mark start component:

```

>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
  
```

Return component or none.

Inherited from `marktools.Mark`

`StemTremolo.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`StemTremolo.tremolo_flags`

Get tremolo flags:

```
>>> stem_tremolo = marktools.StemTremolo(16)
>>> stem_tremolo.tremolo_flags
16
```

Set tremolo flags:

```
>>> stem_tremolo.tremolo_flags = 32
>>> stem_tremolo.tremolo_flags
32
```

Set integer.

Methods

`StemTremolo.attach(start_component)`

Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()

>>> mark.attach(note)
Mark() (c'4)
```

```
>>> mark.start_component
Note("c'4")
```

Return mark.

Inherited from `marktools.Mark`

`StemTremolo.detach()`

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

```
>>> mark.detach()
Mark()
```

```
>>> mark.start_component is None
True
```

Return mark.

Inherited from `marktools.Mark`

Special Methods

```
StemTremolo.__call__(*args)
    Inherited from marktools.Mark
StemTremolo.__delattr__(*args)
    Inherited from marktools.Mark
StemTremolo.__eq__(expr)
StemTremolo.__ge__(arg)
    Abjad objects by default do not implement this method.
    Raise exception.
    Inherited from abctools.AbjadObject
StemTremolo.__gt__(arg)
    Abjad objects by default do not implement this method.
    Raise exception
    Inherited from abctools.AbjadObject
StemTremolo.__le__(arg)
    Abjad objects by default do not implement this method.
    Raise exception.
    Inherited from abctools.AbjadObject
StemTremolo.__lt__(arg)
    Abjad objects by default do not implement this method.
    Raise exception.
    Inherited from abctools.AbjadObject
StemTremolo.__ne__(arg)
    Inherited from marktools.Mark
StemTremolo.__repr__()
    Inherited from marktools.Mark
StemTremolo.__str__()
```

functions

marktools.attach_annotations_to_components_in_expr

```
marktools.attach_annotations_to_components_in_expr(expr, annotations)
    New in version 2.3. Attach annotations to components in expr:
```

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> annotation = marktools.Annotation('foo', 'bar')
>>> marktools.attach_annotations_to_components_in_expr(staff.leaves, [annotation])

>>> for x in staff:
...     print x, marktools.get_annotations_attached_to_component(x)
...
c'8 (Annotation('foo', 'bar')(c'8),)
d'8 (Annotation('foo', 'bar')(d'8),)
e'8 (Annotation('foo', 'bar')(e'8),)
f'8 (Annotation('foo', 'bar')(f'8),)
```

Return none.

`marktools.attach_articulations_to_notes_and_chords_in_expr`

`marktools.attach_articulations_to_notes_and_chords_in_expr(expr, articulations)`

New in version 2.0. Attach *articulations* to notes and chords in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.attach_articulations_to_notes_and_chords_in_expr(staff, list('^.'))

>>> f(staff)
\new Staff {
  c'8 -\marcato -\staccato
  d'8 -\marcato -\staccato
  e'8 -\marcato -\staccato
  f'8 -\marcato -\staccato
}
```

Return none.

`marktools.attach_lilypond_command_marks_to_components_in_expr`

`marktools.attach_lilypond_command_marks_to_components_in_expr(expr, lily-pond_command_marks)`

New in version 2.3. Attach *lilypond_command_marks* to components in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> lilypond_command_mark = marktools.LilyPondCommandMark('stemUp')
>>> marktools.attach_lilypond_command_marks_to_components_in_expr(
...     staff.leaves, [lilypond_command_mark])

>>> f(staff)
\new Staff {
  \stemUp
  c'8
  \stemUp
  d'8
  \stemUp
  e'8
  \stemUp
  f'8
}
```

Return none.

`marktools.attach_lilypond_comments_to_components_in_expr`

`marktools.attach_lilypond_comments_to_components_in_expr(expr, lily-pond_comments)`

New in version 2.3. Attach *lilypond_comments* to components in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> lilypond_comment = marktools.LilyPondComment('foo', 'right')
>>> marktools.attach_lilypond_comments_to_components_in_expr(
...     staff.leaves, [lilypond_comment])
```

```
>>> f(staff)
\new Staff {
  c'8 % foo
  d'8 % foo
  e'8 % foo
  f'8 % foo
}
```

Return none.

marktools.attach_stem_tremolos_to_notes_and_chords_in_expr

marktools.attach_stem_tremolos_to_notes_and_chords_in_expr (*expr*, *stem_tremolos*)

New in version 2.3. Attach *stem_tremolos* to notes and chords in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> stem_tremolo = marktools.StemTremolo(16)
>>> marktools.attach_stem_tremolos_to_notes_and_chords_in_expr(staff, [stem_tremolo])

>>> f(staff)
\new Staff {
  c'8 :16
  d'8 :16
  e'8 :16
  f'8 :16
}
```

Return none.

marktools.detach_annotations_attached_to_component

marktools.detach_annotations_attached_to_component (*component*)

New in version 2.0. Detach annotations attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> marktools.Annotation('annotation 1')(staff[0])
Annotation('annotation 1')(c'8)
>>> marktools.Annotation('annotation 2')(staff[0])
Annotation('annotation 2')(c'8)

>>> f(staff)
\new Staff {
  c'8 (
  d'8
  e'8
  f'8 )
}
```

```
>>> marktools.get_annotations_attached_to_component(staff[0])
(Annotation('annotation 1')(c'8), Annotation('annotation 2')(c'8))

>>> marktools.detach_annotations_attached_to_component(staff[0])
(Annotation('annotation 1'), Annotation('annotation 2'))
```

```
>>> marktools.get_annotations_attached_to_component(staff[0])
()
```

Return tuple or zero or more annotations detached.

marktools.detach_articulations_attached_to_component

marktools.detach_articulations_attached_to_component(*component*)

New in version 2.0. Detach articulations attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> marktools.Articulation('^')(staff[0])
Articulation('^')(c'8)
>>> marktools.Articulation('.')[staff[0])
Articulation('.')[c'8)

>>> f(staff)
\new Staff {
  c'8 -\marcato -\staccato (
  d'8
  e'8
  f'8 )
}

>>> marktools.get_articulations_attached_to_component(staff[0])
(Articulation('^')(c'8), Articulation('.')[c'8))

>>> marktools.detach_articulations_attached_to_component(staff[0])
(Articulation('^'), Articulation('.'))

>>> marktools.get_articulations_attached_to_component(staff[0])
()
```

Return tuple or zero or more articulations detached.

marktools.detach_lilypond_command_marks_attached_to_component

marktools.detach_lilypond_command_marks_attached_to_component(*component*, *command_name=None*)

New in version 2.0. Detach LilyPond command marks attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> marktools.LilyPondCommandMark('slurDotted')(staff[0])
LilyPondCommandMark('slurDotted')(c'8)
>>> marktools.LilyPondCommandMark('slurUp')(staff[0])
LilyPondCommandMark('slurUp')(c'8)

>>> f(staff)
\new Staff {
  \slurDotted
  \slurUp
  c'8 (
  d'8
  e'8
```

```

        f'8 )
    }

>>> marktools.detach_lilypond_command_marks_attached_to_component(staff[0])
(LilyPondCommandMark('slurDotted'), LilyPondCommandMark('slurUp'))

>>> f(staff)
\new Staff {
    c'8 (
    d'8
    e'8
    f'8 )
}

```

Return tuple of zero or more marks detached.

marktools.detach_lilypond_comments_attached_to_component

marktools.detach_lilypond_comments_attached_to_component (*component*)

New in version 2.0. Detach LilyPond comments attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> marktools.LilyPondComment('comment 1')(staff[0])
LilyPondComment('comment 1')(c'8)
>>> marktools.LilyPondComment('comment 2')(staff[0])
LilyPondComment('comment 2')(c'8)

>>> f(staff)
\new Staff {
    % comment 1
    % comment 2
    c'8 (
    d'8
    e'8
    f'8 )
}

>>> marktools.detach_lilypond_comments_attached_to_component(staff[0])
(LilyPondComment('comment 1'), LilyPondComment('comment 2'))

>>> f(staff)
\new Staff {
    c'8 (
    d'8
    e'8
    f'8 )
}

>>> marktools.get_lilypond_comments_attached_to_component(staff[0])
()

```

Return tuple of zero or more LilyPond comments.

marktools.detach_marks_attached_to_component

marktools.**detach_marks_attached_to_component** (*component*)

New in version 2.0. Detach marks attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> marktools.Arterulation('^')(staff[0])
Arterulation('^')(c'8)
>>> marktools.LilyPondComment('comment 1')(staff[0])
LilyPondComment('comment 1')(c'8)
>>> marktools.LilyPondCommandMark('slurUp')(staff[0])
LilyPondCommandMark('slurUp')(c'8)

>>> f(staff)
\new Staff {
  % comment 1
  \slurUp
  c'8 -\marcato (
  d'8
  e'8
  f'8 )
}

>>> for mark in marktools.get_marks_attached_to_component(staff[0]):
...     mark
...
Arterulation('^')(c'8)
LilyPondComment('comment 1')(c'8)
LilyPondCommandMark('slurUp')(c'8)

>>> for mark in marktools.detach_marks_attached_to_component(staff[0]):
...     mark
...
Arterulation('^')
LilyPondComment('comment 1')
LilyPondCommandMark('slurUp')

>>> marktools.get_marks_attached_to_component(staff[0])
()
```

Return tuple or zero or more marks detached.

marktools.detach_marks_attached_to_components_in_expr

marktools.**detach_marks_attached_to_components_in_expr** (*expr*)

New in version 2.9. Detach marks attached to components in *expr*:

```
>>> staff = Staff("c'4 \staccato d' \marcato e' \staccato f' \marcato")

>>> f(staff)
\new Staff {
  c'4 -\staccato
  d'4 -\marcato
  e'4 -\staccato
  f'4 -\marcato
}
```

```

>>> for mark in marktools.detach_marks_attached_to_components_in_expr(staff):
...     mark
...
Articulation('staccato')
Articulation('marcato')
Articulation('staccato')
Articulation('marcato')

>>> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
}

```

Return tuple of zero or more detached marks.

marktools.detach_noncontext_marks_attached_to_component

marktools.detach_noncontext_marks_attached_to_component (*component*)

New in version 2.3. Detach noncontext marks attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.TimeSignatureMark((2, 4))(staff[0])
TimeSignatureMark((2, 4))(c'8)
>>> marktools.Aarticulation('staccato')(staff[0])
Articulation('staccato')(c'8)

>>> f(staff)
\new Staff {
    \time 2/4
    c'8 -\staccato
    d'8
    e'8
    f'8
}

>>> marktools.detach_noncontext_marks_attached_to_component(staff[0])
(Articulation('staccato'),)

>>> f(staff)
\new Staff {
    \time 2/4
    c'8
    d'8
    e'8
    f'8
}

```

Return tuple of noncontext marks.

marktools.detach_stem_tremolos_attached_to_component

marktools.detach_stem_tremolos_attached_to_component (*component*)

New in version 2.0. Detach stem tremolos attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.StemTremolo(16)(staff[0])
StemTremolo(16)(c'8)

>>> f(staff)
\new Staff {
  c'8 :16
  d'8
  e'8
  f'8
}

>>> marktools.get_stem_tremolos_attached_to_component(staff[0])
(StemTremolo(16)(c'8),)

>>> marktools.detach_stem_tremolos_attached_to_component(staff[0])
(StemTremolo(16),)

>>> marktools.get_stem_tremolos_attached_to_component(staff[0])
()
```

Return tuple or zero or more stem tremolos detached.

marktools.get_annotation_attached_to_component

marktools.get_annotation_attached_to_component (*component*)

New in version 2.0. Get exactly one annotation attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.Annotation('special information')(staff[0])
Annotation('special information')(c'8)

>>> f(staff)
\new Staff {
  c'8
  d'8
  e'8
  f'8
}

>>> marktools.get_annotation_attached_to_component(staff[0])
Annotation('special information')(c'8)
```

Return one annotation.

Raise missing mark error when no annotation is attached.

Raise extra mark error when more than one annotation is attached.

marktools.get_annotations_attached_to_component

marktools.get_annotations_attached_to_component (*component*)

New in version 2.0. Get annotations attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.Annotation('annotation 1')(staff[0])
```



```

Annotation('annotation 1')(c'8)
>>> marktools.Annotation('annotation 2')(staff[0])
Annotation('annotation 2')(c'8)

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

>>> marktools.get_annotations_attached_to_component(staff[0])
(Annotation('annotation 1')(c'8), Annotation('annotation 2')(c'8))

```

Return tuple of zero or more annotations.

marktools.get_articulation_attached_to_component

marktools.get_articulation_attached_to_component (*component*)

New in version 2.0. Get exactly one articulation attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.Articulation('staccato')(staff[0])
Articulation('staccato')(c'8)

>>> f(staff)
\new Staff {
    c'8 -\staccato
    d'8
    e'8
    f'8
}

>>> marktools.get_articulation_attached_to_component(staff[0])
Articulation('staccato')(c'8)

```

Return one articulation.

Raise missing mark error when no articulation is attached.

Raise extra mark error when more than one articulation is attached.

marktools.get_articulations_attached_to_component

marktools.get_articulations_attached_to_component (*component*)

New in version 2.0. Get articulations attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.Articulation('staccato')(staff[0])
Articulation('staccato')(c'8)
>>> marktools.Articulation('marcato')(staff[0])
Articulation('marcato')(c'8)

>>> f(staff)
\new Staff {
    c'8 -\marcato -\staccato
}

```

```

        d'8
        e'8
        f'8
    }

>>> marktools.get_articulations_attached_to_component(staff[0])
(Articulation('staccato')(c'8), Articulation('marcato')(c'8))

```

Return tuple of zero or more articulations.

marktools.get_lilypond_command_mark_attached_to_component

marktools.get_lilypond_command_mark_attached_to_component (*component*)

New in version 2.0. Get exactly one LilyPond command mark attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.LilyPondCommandMark('stemUp')(staff[0])
LilyPondCommandMark('stemUp')(c'8)

>>> f(staff)
\new Staff {
  \stemUp
  c'8
  d'8
  e'8
  f'8
}

>>> marktools.get_lilypond_command_mark_attached_to_component(staff[0])
LilyPondCommandMark('stemUp')(c'8)

```

Return one LilyPond command mark.

Raise missing mark error when no LilyPond command mark is attached.

Raise extra mark error when more than one LilyPond command mark is attached.

marktools.get_lilypond_command_marks_attached_to_component

marktools.get_lilypond_command_marks_attached_to_component (*component*, *command_name=None*)

New in version 2.0. Get LilyPond command marks attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> marktools.LilyPondCommandMark('slurDotted')(staff[0])
LilyPondCommandMark('slurDotted')(c'8)
>>> marktools.LilyPondCommandMark('slurUp')(staff[0])
LilyPondCommandMark('slurUp')(c'8)

>>> f(staff)
\new Staff {
  \slurDotted
  \slurUp
  c'8 (
  d'8
  e'8

```

```
f'8 )
}

>>> marktools.get_lilypond_command_marks_attached_to_component(staff[0])
(LilyPondCommandMark('slurDotted')(c'8), LilyPondCommandMark('slurUp')(c'8))
```

Return tuple of zero or more marks.

marktools.get_lilypond_comment_attached_to_component

marktools.get_lilypond_comment_attached_to_component (*component*)

New in version 2.0. Get exactly one LilyPond comment attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.LilyPondComment('comment')(staff[0])
LilyPondComment('comment')(c'8)

>>> f(staff)
\new Staff {
  % comment
  c'8
  d'8
  e'8
  f'8
}

>>> marktools.get_lilypond_comment_attached_to_component(staff[0])
LilyPondComment('comment')(c'8)
```

Return one LilyPond comment.

Raise missing mark error when no LilyPond comment is attached.

Raise extra mark error when more than one LilyPond comment is attached.

marktools.get_lilypond_comments_attached_to_component

marktools.get_lilypond_comments_attached_to_component (*component*)

New in version 2.0. Get LilyPond comments attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> marktools.LilyPondComment('comment 1')(staff[0])
LilyPondComment('comment 1')(c'8)
>>> marktools.LilyPondComment('comment 2')(staff[0])
LilyPondComment('comment 2')(c'8)

>>> f(staff)
\new Staff {
  % comment 1
  % comment 2
  c'8 (
  d'8
  e'8
  f'8 )
}
```

```
>>> marktools.get_lilypond_comments_attached_to_component(staff[0])
(LilyPondComment('comment 1')(c'8), LilyPondComment('comment 2')(c'8))
```

Return tuple of zero or more LilyPond comments.

marktools.get_mark_attached_to_component

marktools.get_mark_attached_to_component (*component*)
 New in version 2.0. Get exactly one mark attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.Mark()(staff[0])
Mark()(c'8)

>>> f(staff)
\new Staff {
  c'8
  d'8
  e'8
  f'8
}

>>> marktools.get_mark_attached_to_component(staff[0])
Mark()(c'8)
```

Return one mark.

Raise missing mark error when no mark is attached.

Raise extra mark error when more than one mark is attached.

marktools.get_marks_attached_to_component

marktools.get_marks_attached_to_component (*component*)
 New in version 2.0. Get all marks attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> comment_mark = marktools.LilyPondComment('beginning of note content')(staff[0])
>>> marktools.LilyPondCommandMark('slurDotted')(staff[0])
LilyPondCommandMark('slurDotted')(c'8)

>>> f(staff)
\new Staff {
  % beginning of note content
  \slurDotted
  c'8 (
  d'8
  e'8
  f'8 )
}

>>> marktools.get_marks_attached_to_component(staff[0])
(LilyPondComment('beginning of note content')(c'8), LilyPondCommandMark('slurDotted')(c'8))
```

Return tuple of zero or more marks. Changed in version 2.0: re-named `marktools.get_all_marks_attached_to_component()` to `marktools.get_marks_attached_to_component()`.

`marktools.get_marks_attached_to_components_in_expr`

`marktools.get_marks_attached_to_components_in_expr(expr)`

New in version 2.9. Get marks attached to components in *expr*:

```
>>> staff = Staff(r"c'4 \pp d' \staccato e' \ff f' \staccato")

\new Staff {
  c'4 \pp
  d'4 -\staccato
  e'4 \ff
  f'4 -\staccato
}

>>> for mark in marktools.get_marks_attached_to_components_in_expr(staff):
...     mark
...
DynamicMark('pp')(c'4)
Articulation('staccato')(d'4)
DynamicMark('ff')(e'4)
Articulation('staccato')(f'4)
```

Return tuple of zero or more marks.

`marktools.get_noncontext_mark_attached_to_component`

`marktools.get_noncontext_mark_attached_to_component(component)`

New in version 2.0. Get exactly one `noncontext_mark` attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.Articulation('staccato')(staff[0])
Articulation('staccato')(c'8)

>>> f(staff)
\new Staff {
  c'8 -\staccato
  d'8
  e'8
  f'8
}

>>> marktools.get_noncontext_mark_attached_to_component(staff[0])
Articulation('staccato')(c'8)
```

Return one `noncontext_mark`.

Raise missing mark error when no `noncontext_mark` is attached.

Raise extra mark error when more than one `noncontext_mark` is attached.

marktools.get_noncontext_marks_attached_to_component

`marktools.get_noncontext_marks_attached_to_component` (*component*)

New in version 2.0. Get noncontext marks attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> contexttools.TimeSignatureMark((2, 4))(staff[0])
TimeSignatureMark((2, 4))(c'8)
>>> marktools.Articulation('staccato')(staff[0])
Articulation('staccato')(c'8)

>>> f(staff)
\new Staff {
  \time 2/4
  c'8 -\staccato
  d'8
  e'8
  f'8
}

>>> marktools.get_noncontext_marks_attached_to_component(staff[0])
(Articulation('staccato')(c'8),)
```

Return tuple of zero or more marks.

marktools.get_stem_tremolo_attached_to_component

`marktools.get_stem_tremolo_attached_to_component` (*component*)

New in version 2.0. Get stem tremolo attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.StemTremolo(16)(staff[0])
StemTremolo(16)(c'8)

>>> f(staff)
\new Staff {
  c'8 :16
  d'8
  e'8
  f'8
}

>>> marktools.get_stem_tremolo_attached_to_component(staff[0])
StemTremolo(16)(c'8)
```

Raise missing mark error when no stem tremolo attaches to *component*.

Raise extra mark error when more than one stem tremolo attaches to *component*.

Return stem tremolo.

marktools.get_stem_tremolos_attached_to_component

`marktools.get_stem_tremolos_attached_to_component` (*component*, *tremolo_flags=None*)

New in version 2.3. Get stem tremolos attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.StemTremolo(16)(staff[0])
StemTremolo(16)(c'8)

>>> f(staff)
\new Staff {
    c'8 :16
    d'8
    e'8
    f'8
}

>>> marktools.get_stem_tremolos_attached_to_component(staff[0])
(StemTremolo(16)(c'8),)
```

Return tuple of zero or more stem tremolos.

marktools.get_value_of_annotation_attached_to_component

`marktools.get_value_of_annotation_attached_to_component` (*component*, *name*, *default_value=None*)

New in version 2.0. Get value of annotation with *name* attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> marktools.Annotation('special dictionary', {})(staff[0])
Annotation('special dictionary', {})(c'8)

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

>>> marktools.get_value_of_annotation_attached_to_component(staff[0], 'special dictionary')
{}
```

Return arbitrary value of annotation.

Return *default_value* when no annotation with *name* is attached.

Raise extra mark error when more than one annotation with *name* is attached.

marktools.is_component_with_annotation_attached

`marktools.is_component_with_annotation_attached` (*expr*, *annotation_name=None*, *annotation_value=None*)

New in version 2.3. True when *expr* is component with annotation attached:

```
>>> note = Note("c'4")
>>> marktools.Annotation('foo', 'bar')(note)
Annotation('foo', 'bar')(c'4)

>>> marktools.is_component_with_annotation_attached(note)
True
```

False otherwise:

```
>>> note = Note("c'4")

>>> marktools.is_component_with_annotation_attached(note)
False
```

Return boolean.

marktools.is_component_with_articulation_attached

`marktools.is_component_with_articulation_attached(expr, articulation_name=None)`

New in version 2.3. True when *expr* is component with articulation attached:

```
>>> note = Note("c'4")
>>> marktools.Articulation('staccato')(note)
Articulation('staccato')(c'4)

>>> marktools.is_component_with_articulation_attached(note)
True
```

False otherwise:

```
>>> note = Note("c'4")

>>> marktools.is_component_with_articulation_attached(note)
False
```

Return boolean.

marktools.is_component_with_lilypond_command_mark_attached

`marktools.is_component_with_lilypond_command_mark_attached(expr, command_name=None)`

New in version 2.0. True when *expr* is component with LilyPond command mark attached:

```
>>> note = Note("c'4")
>>> marktools.LilyPondCommandMark('stemUp')(note)
LilyPondCommandMark('stemUp')(c'4)

>>> marktools.is_component_with_lilypond_command_mark_attached(note)
True
```

False otherwise:

```
>>> note = Note("c'4")

>>> marktools.is_component_with_lilypond_command_mark_attached(note)
False
```

Return boolean.

marktools.is_component_with_lilypond_comment_attached

`marktools.is_component_with_lilypond_comment_attached(expr, comment_contents_string=None)`

New in version 2.3. True when *expr* is component with LilyPond comment mark attached:


```
>>> note = Note("c'4")
>>> marktools.LilyPondComment('comment here')(note)
LilyPondComment('comment here')(c'4)

>>> marktools.is_component_with_lilypond_comment_attached(note)
True
```

False otherwise:

```
>>> note = Note("c'4")

>>> marktools.is_component_with_lilypond_comment_attached(note)
False
```

Return boolean.

marktools.is_component_with_mark_attached

marktools.is_component_with_mark_attached(*expr*)

New in version 2.3. True when *expr* is component with mark attached:

```
>>> note = Note("c'4")
>>> marktools.Mark()(note)
Mark()(c'4)

>>> marktools.is_component_with_mark_attached(note)
True
```

False otherwise:

```
>>> note = Note("c'4")

>>> marktools.is_component_with_mark_attached(note)
False
```

Return boolean.

marktools.is_component_with_noncontext_mark_attached

marktools.is_component_with_noncontext_mark_attached(*expr*)

New in version 2.3. True when *expr* is component with noncontext mark attached:

```
>>> note = Note("c'4")
>>> marktools.Articulation('staccato')(note)
Articulation('staccato')(c'4)

>>> marktools.is_component_with_noncontext_mark_attached(note)
True
```

False otherwise:

```
>>> note = Note("c'4")

>>> marktools.is_component_with_noncontext_mark_attached(note)
False
```

Return boolean.

marktools.is_component_with_stem_tremolo_attached

`marktools.is_component_with_stem_tremolo_attached(expr)`

New in version 2.3. True when *expr* is component with LilyPond command mark attached:

```
>>> note = Note("c'4")
>>> marktools.StemTremolo(16)(note)
StemTremolo(16)(c'4)

>>> marktools.is_component_with_stem_tremolo_attached(note)
True
```

False otherwise:

```
>>> note = Note("c'4")

>>> marktools.is_component_with_stem_tremolo_attached(note)
False
```

Return boolean.

marktools.move_marks

`marktools.move_marks(donor, recipient)`

Move marks from *donor* component to *recipient* component:

```
>>> staff = Staff(r'\clef "bass" c \staccato d e f')

>>> f(staff)
\new Staff {
  \clef "bass"
  c4 -\staccato
  d4
  e4
  f4
}

>>> marktools.move_marks(staff[0], staff[2])
[Articulation('staccato')(e4), ClefMark('bass')(e4)]

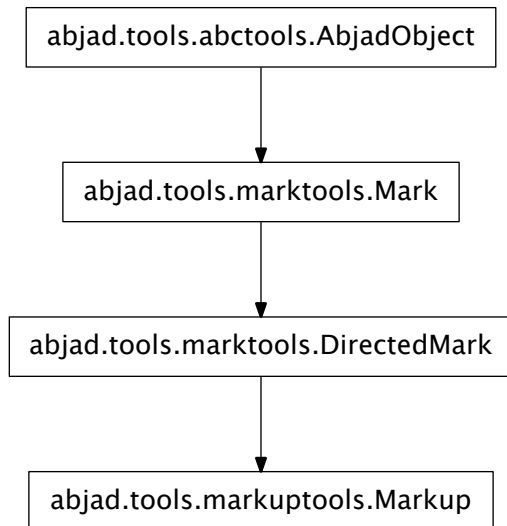
>>> f(staff)
\new Staff {
  c4
  d4
  \clef "bass"
  e4 -\staccato
  f4
}
```

Return list of marks moved.

markuptools

concrete classes

markuptools.Markup



class markuptools.**Markup** (*argument*, *direction=None*, *markup_name=None*)
 Abjad model of LilyPond markup.

Initialize from string:

```

>>> markup = markuptools.Markup(r'\bold { "This is markup text." }')

>>> markup
Markup((MarkupCommand('bold', ['This is markup text.']),))

>>> f(markup)
\markup { \bold { "This is markup text." } }
  
```

Initialize any markup from existing markup:

```

>>> markup_1 = markuptools.Markup('foo', direction=Up)
>>> markup_2 = markuptools.Markup(markup_1, direction=Down)

>>> f(markup_1)
^ \markup { foo }

>>> f(markup_2)
_ \markup { foo }
  
```

Attach markup to score components like this:

```
>>> note = Note("c'4")

>>> markup = markuptools.Markup(r'\bold { "This is markup text." }')

>>> markup(note)
Markup((MarkupCommand('bold', ['This is markup text.']),)(c'4)

>>> f(note)
c'4 - \markup { \bold { "This is markup text." } }
```

Set *direction* to Up, Down, 'neutral', '^', '_', '-' or None.

Markup objects are immutable.

Read-only Properties

Markup.contents

Read-only tuple of contents of markup:

```
>>> markup = markuptools.Markup(r'\bold { "This is markup text." }')
>>> markup.contents
(MarkupCommand('bold', ['This is markup text.']),)
```

Return string

Markup.indented_lilypond_format

Read-only indented LilyPond format of markup:

```
>>> markup = markuptools.Markup(r'\bold { "This is markup text." }')
>>> print markup.indented_lilypond_format
\markup {
  \bold {
    "This is markup text."
  }
}
```

Return string.

Markup.lilypond_format

Read-only LilyPond format of markup:

```
>>> markup = markuptools.Markup(r'\bold { "This is markup text." }')
>>> markup.lilypond_format
'\markup { \bold { "This is markup text." } }'
```

Return string.

Markup.markup_name

Read-only name of markup:

```
>>> markup = markuptools.Markup(
...     r'\bold { allegro ma non troppo }', markup_name='non troppo')

>>> markup.markup_name
'non troppo'
```

Return string or none.

Markup.start_component

Read-only reference to mark start component:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")
```

Return component or none.

Inherited from `marktools.Mark`

Markup.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Markup.**direction**

Inherited from `marktools.DirectedMark`

Methods

Markup.**attach**(*start_component*)

Attach mark to *start_component*:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()

>>> mark.attach(note)
Mark()(c'4)

>>> mark.start_component
Note("c'4")
```

Return mark.

Inherited from `marktools.Mark`

Markup.**detach**()

Detach mark:

```
>>> note = Note("c'4")
>>> mark = marktools.Mark()(note)

>>> mark.start_component
Note("c'4")

>>> mark.detach()
Mark()

>>> mark.start_component is None
True
```

Return mark.

Inherited from `marktools.Mark`

Special Methods

Markup.__call__(*args)

Inherited from marktools.Mark

Markup.__delattr__(*args)

Inherited from marktools.Mark

Markup.__eq__(expr)

Markup.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

Markup.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

Markup.__hash__()

Markup.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

Markup.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

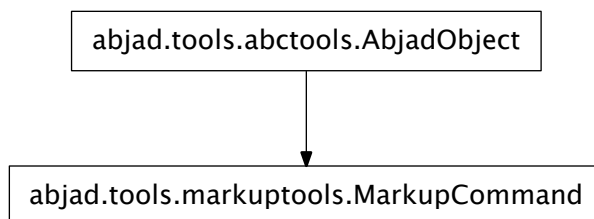
Markup.__ne__(expr)

Markup.__repr__()

Inherited from marktools.Mark

Markup.__str__()

markuptools.MarkupCommand



class `markuptools.MarkupCommand(command, *args)`

Abjad model of a LilyPond markup command:

```
>>> circle = markuptools.MarkupCommand('draw-circle', 2.5, 0.1, False)
>>> square = markuptools.MarkupCommand('rounded-box', 'hello?')
>>> line = markuptools.MarkupCommand('line', [square, 'wow!'])
>>> rotate = markuptools.MarkupCommand('rotate', 60, line)
>>> combine = markuptools.MarkupCommand('combine', rotate, circle)

>>> print combine
\combine \rotate #60 \line { \rounded-box hello? wow! } \draw-circle #2.5 #0.1 ##f
```

Insert markup command in markup to attach to score components:

```
>>> note = Note("c'4")

>>> markup = markuptools.Markup(combine)

>>> markup = markup(note)

>>> f(note)
c'4
- \markup {
  \combine
    \rotate
      #60
      \line
        {
          \rounded-box
            hello?
            wow!
        }
    \draw-circle
      #2.5
      #0.1
      ##f
}
```

Markup commands are immutable.

Read-only Properties

`MarkupCommand.args`

Read-only tuple of markup command arguments.

`MarkupCommand.command`

Read-only string of markup command command-name.

`MarkupCommand.lilypond_format`

Read-only format of markup command:

```
>>> markup_command = markuptools.MarkupCommand('draw-circle', 2.5, 0.1, False)
>>> markup_command.lilypond_format
'\draw-circle #2.5 #0.1 ##f'
```

Returns string.

`MarkupCommand.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`MarkupCommand.__eq__(other)`

`MarkupCommand.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MarkupCommand.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MarkupCommand.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MarkupCommand.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MarkupCommand.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

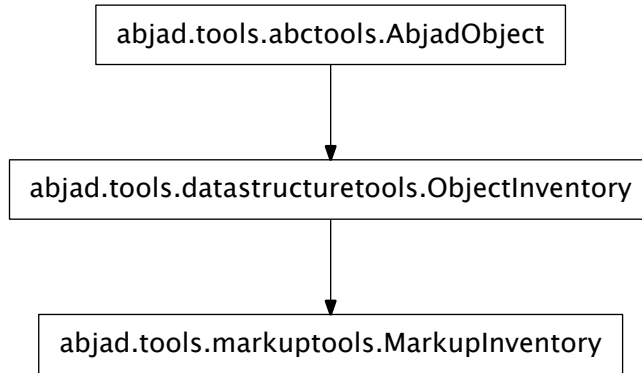
Return boolean.

Inherited from `abctools.AbjadObject`

`MarkupCommand.__repr__()`

`MarkupCommand.__str__()`

markuptools.MarkupInventory



class markuptools.**MarkupInventory** (*tokens=None, name=None*)

New in version 2.8. Abjad model of an ordered list of markup:

```
>>> inventory = markuptools.MarkupInventory(['foo', 'bar'])
```

```
>>> inventory
MarkupInventory([Markup(('foo',)), Markup(('bar',))])
```

Markup inventories implement the list interface and are mutable.

Read-only Properties

MarkupInventory.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from abctools.AbjadObject

Read/write Properties

MarkupInventory.**name**

Read / write name of inventory.

Inherited from datastructuretools.ObjectInventory

Methods

MarkupInventory.**append** (*token*)

Change *token* to item and append.

Inherited from datastructuretools.ObjectInventory

MarkupInventory.**count** (*value*) → integer – return number of occurrences of value

Inherited from `__builtin__.list`

MarkupInventory.**extend** (*tokens*)

Change *tokens* to items and extend.

Inherited from `datastructuretools.ObjectInventory`

`MarkupInventory.index(value[, start[, stop]])` → integer – return first index of value.
 Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`MarkupInventory.insert()`
`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`MarkupInventory.pop([index])` → item – remove and return item at index (default last).
 Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`MarkupInventory.remove()`
`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`MarkupInventory.reverse()`
`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`MarkupInventory.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`MarkupInventory.__add__()`
`x.__add__(y) <==> x+y`

Inherited from `__builtin__.list`

`MarkupInventory.__contains__(token)`
 Inherited from `datastructuretools.ObjectInventory`

`MarkupInventory.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Inherited from `__builtin__.list`

`MarkupInventory.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
 Use of negative indices is not supported.

Inherited from `__builtin__.list`

`MarkupInventory.__eq__()`
`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.list`

`MarkupInventory.__ge__()`
`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.list`

`MarkupInventory.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.list`

`MarkupInventory.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

`MarkupInventory.__gt__()`
`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.list`

`MarkupInventory.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Inherited from `__builtin__.list`

`MarkupInventory.__imul__()`
`x.__imul__(y) <==> x*=y`

Inherited from `__builtin__.list`

`MarkupInventory.__iter__()` <==> `iter(x)`
 Inherited from `__builtin__.list`

`MarkupInventory.__le__()`
`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.list`

`MarkupInventory.__len__()` <==> `len(x)`
 Inherited from `__builtin__.list`

`MarkupInventory.__lt__()`
`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.list`

`MarkupInventory.__mul__()`
`x.__mul__(n) <==> x*n`

Inherited from `__builtin__.list`

`MarkupInventory.__ne__()`
`x.__ne__(y) <==> x!=y`

Inherited from `__builtin__.list`

`MarkupInventory.__repr__()`
 Inherited from `datastructuretools.ObjectInventory`

`MarkupInventory.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Inherited from `__builtin__.list`

`MarkupInventory.__rmul__()`
`x.__rmul__(n) <==> n*x`

Inherited from `__builtin__.list`

`MarkupInventory.__setitem__()`
`x.__setitem__(i, y) <==> x[i]=y`

Inherited from `__builtin__.list`

`MarkupInventory.__setslice__()`
`x.__setslice__(i, j, y) <==> x[i:j]=y`
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

functions

`markuptools.all_are_markup`

`markuptools.all_are_markup(expr)`
 New in version 2.6. True when *expr* is a sequence of Abjad markup:

```
>>> markup = markuptools.Markup('some text')

>>> markuptools.all_are_markup([markup])
True
```

True when *expr* is an empty sequence:

```
>>> markuptools.all_are_markup([])
True
```

Otherwise false:

```
>>> markuptools.all_are_markup('foo')
False
```

Return boolean.

`markuptools.combine_markup_commands`

`markuptools.combine_markup_commands(*commands)`
 Combine MarkupCommand and/or string objects.

LilyPond's 'combine' markup command can only take two arguments, so in order to combine more than two stencils, a cascade of 'combine' commands must be employed. *combine_markup_commands* simplifies this process.

```
>>> from abjad.tools.schemetools import SchemePair

>>> markup_a = markuptools.MarkupCommand('draw-circle', 4, 0.4, False)
>>> markup_b = markuptools.MarkupCommand(
...     'filled-box',
...     schemetools.SchemePair(-4, 4),
...     schemetools.SchemePair(-0.5, 0.5), 1)
>>> markup_c = "some text"

>>> markup = markuptools.combine_markup_commands(markup_a, markup_b, markup_c)
>>> result = markup.lilypond_format

>>> print result
\combine \combine \draw-circle #4 #0.4 ##f
    \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1 "some text"
```

Returns a markup command instance, or a string if that was the only argument.

markuptools.get_down_markup_attached_to_component

`markuptools.get_down_markup_attached_to_component` (*component*)

New in version 2.0. Get down-markup attached to component:

```
>>> chord = Chord([-11, 2, 5], (1, 4))

>>> markuptools.Markup('UP', Up)(chord)
Markup(('UP',), direction=Up)(<cs d' f'>4)

>>> markuptools.Markup('DOWN', Down)(chord)
Markup(('DOWN',), direction=Down)(<cs d' f'>4)

>>> markuptools.get_down_markup_attached_to_component(chord)
(Markup(('DOWN',), direction=Down)(<cs d' f'>4),)
```

Return tuple of zero or more markup objects.

markuptools.get_markup_attached_to_component

`markuptools.get_markup_attached_to_component` (*component*)

New in version 2.0. Get markup attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff[:])

>>> markuptools.Markup('foo')(staff[0])
Markup(('foo',)) (c'8)

>>> markuptools.Markup('bar')(staff[0])
Markup(('bar',)) (c'8)

>>> f(staff)
\new Staff {
  c'8 (
    - \markup {
      \column
      {
        foo
        bar
      }
    )
  d'8
  e'8
  f'8 )
}

>>> markuptools.get_markup_attached_to_component(staff[0])
(Markup(('foo',)) (c'8), Markup(('bar',)) (c'8))
```

Return tuple of zero or more markup objects.

markuptools.get_up_markup_attached_to_component

`markuptools.get_up_markup_attached_to_component` (*component*)

New in version 2.0. Get up-markup attached to component:

```
>>> chord = Chord([-11, 2, 5], (1, 4))

>>> markuptools.Markup('UP', Up)(chord)
Markup(('UP',), direction=Up)(<cs d' f'>4)

>>> markuptools.Markup('DOWN', Down)(chord)
Markup(('DOWN',), direction=Down)(<cs d' f'>4)

>>> markuptools.get_up_markup_attached_to_component(chord)
(Markup(('UP',), direction=Up)(<cs d' f'>4),)
```

Return tuple of zero or more markup objects.

markuptools.make_big_centered_page_number_markup

markuptools.**make_big_centered_page_number_markup**(text=None)
 New in version 1.1. Make big centered page number markup:

```
>>> markup = markuptools.make_big_centered_page_number_markup()

>>> print markup.indented_lilypond_format
\markup {
  \fill-line
  {
    \bold
    \fontsize
    #3
    \concat
    {
      \on-the-fly
      print
      page-number-check-first
      \fromproperty
      page
      :page-number-string
    }
  }
}
```

Return markup. Changed in version 2.0: renamed `markuptools.big_centered_page_number()` to `markuptools.make_big_centered_page_number_markup()`.

markuptools.make_blank_line_markup

markuptools.**make_blank_line_markup**()
 New in version 2.9. Make blank line markup:

```
>>> markup = markuptools.make_blank_line_markup()

>>> markup
Markup((MarkupCommand('fill-line', [' ']),))

>>> f(markup)
\markup { \fill-line { " " } }
```

Return markup.

markuptools.make_centered_title_markup

markuptools.**make_centered_title_markup**(*title*, *font_name*='Times', *font_size*=18)

New in version 2.9. Make centered *title* markup:

```
>>> markup = markuptools.make_centered_title_markup('String Quartet')
```

```
>>> print markup.indented_lilypond_format
\markup {
  \column
  {
    \center-align
    {
      \override
        #'(font - name Times)
        \fontsize
          #18
          {
            " "
            " "
            " "
            " "
            " "
            \line
            {
              "String Quartet"
            }
            " "
            " "
            " "
          }
        }
      }
    }
  }
}
```

Return markup.

markuptools.make_vertically_adjusted_composer_markup

markuptools.**make_vertically_adjusted_composer_markup**(*composer*, *font_name*='Times',
font_size=3, *space_above*=20,
space_right=0)

New in version 2.9. Make vertically adjusted *composer* markup:

```
>>> markup = markuptools.make_vertically_adjusted_composer_markup('Josquin Desprez')
```

```
>>> print markup.indented_lilypond_format
\markup {
  \override
    #'(font - name Times)
    {
      \hspace
        #0
      \raise
        #-20
        \fontsize
          #3
```

```

        "Josquin Desprez"
    \hspace
    #0
}
}

```

Return markup.

markuptools.remove_markup_attached_to_component

markuptools.remove_markup_attached_to_component (*component*)

New in version 2.0. Remove markup attached to *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> slur = spannertools.SlurSpanner(staff[:])
>>> markuptools.Markup('foo')(staff[0])
Markup(('foo',))(c'8)
>>> markuptools.Markup('bar')(staff[0])
Markup(('bar',))(c'8)

>>> f(staff)
\new Staff {
  c'8 (
    - \markup {
      \column
      {
        foo
        bar
      }
    )
  d'8
  e'8
  f'8 )
}

>>> markuptools.remove_markup_attached_to_component(staff[0])
(Markup(('foo',)), Markup(('bar',)))

>>> f(staff)
\new Staff {
  c'8 (
  d'8
  e'8
  f'8 )
}

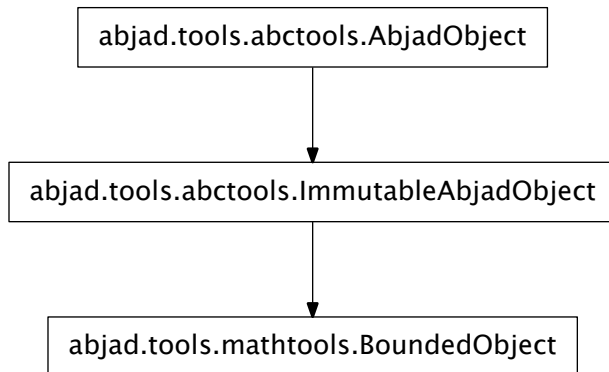
```

Return tuple of zero or more markup objects.

mathtools

abstract classes

mathtools.BoundedObject



class `mathtools.BoundedObject` (**args*, ***kwargs*)

New in version 2.10. Bounded object mix-in.

Bounded objects are immutable.

Read-only Properties

`BoundedObject.is_closed`

`BoundedObject.is_half_closed`

`BoundedObject.is_half_open`

`BoundedObject.is_open`

`BoundedObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`BoundedObject.is_left_closed`

`BoundedObject.is_left_open`

`BoundedObject.is_right_closed`

`BoundedObject.is_right_open`

Special Methods

`BoundedObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`BoundedObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BoundedObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BoundedObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BoundedObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BoundedObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`BoundedObject.__repr__()`

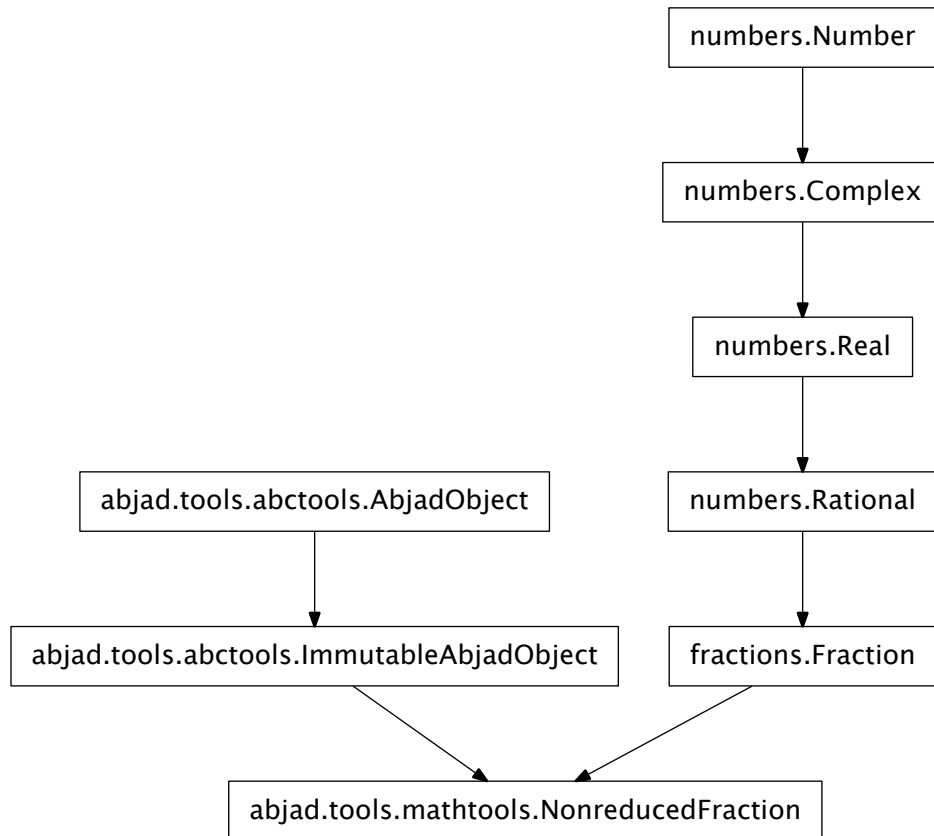
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

concrete classes

mathtools.NonreducedFraction



class `mathtools.NonreducedFraction(*args, **kwargs)`

New in version 2.9. Initialize with an integer numerator and integer denominator:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.NonreducedFraction(3, 6)
NonreducedFraction(3, 6)
```

Initialize with only an integer denominator:

```
>>> mathtools.NonreducedFraction(3)
NonreducedFraction(3, 1)
```

Initialize with an integer pair:

```
>>> mathtools.NonreducedFraction((3, 6))
NonreducedFraction(3, 6)
```

Initialize with an integer singleton:

```
>>> mathtools.NonreducedFraction((3,))
NonreducedFraction(3, 1)
```

Similar to built-in fraction except that numerator and denominator do not reduce.

Nonreduced fractions inherit from built-in fraction:

```
>>> isinstance(mathtools.NonreducedFraction(3, 6), Fraction)
True
```

Nonreduced fractions are numbers:

```
>>> import numbers

>>> isinstance(mathtools.NonreducedFraction(3, 6), numbers.Number)
True
```

Nonreduced fractions are immutable.

Read-only Properties

NonreducedFraction.denominator

Read-only denominator of nonreduced fraction:

```
>>> fraction = mathtools.NonreducedFraction(-6, 3)

abajd> fraction.denominator
3
```

Return positive integer.

NonreducedFraction.imag

Real numbers have no imaginary component.

Inherited from `numbers.Real`

NonreducedFraction.numerator

Read-only numerator of nonreduced fraction:

```
>>> fraction = mathtools.NonreducedFraction(-6, 3)

abajd> fraction.numerator
3
```

Return integer.

NonreducedFraction.pair

Read only pair of nonreduced fraction numerator and denominator:

```
>>> fraction = mathtools.NonreducedFraction(-6, 3)

>>> fraction.pair
(-6, 3)
```

Return integer pair.

NonreducedFraction.real

Real numbers are their real component.

Inherited from `numbers.Real`

`NonreducedFraction.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NonreducedFraction.conjugate()`

Conjugate is a no-op for Reals.

Inherited from `numbers.Real`

classmethod `NonreducedFraction.from_decimal(dec)`

Converts a finite Decimal instance to a rational number, exactly.

Inherited from `fractions.Fraction`

classmethod `NonreducedFraction.from_float(f)`

Converts a finite float to a rational number, exactly.

Beware that `Fraction.from_float(0.3) != Fraction(3, 10)`.

Inherited from `fractions.Fraction`

`NonreducedFraction.limit_denominator(max_denominator=1000000)`

Closest Fraction to self with denominator at most `max_denominator`.

```
>>> Fraction('3.141592653589793').limit_denominator(10)
Fraction(22, 7)
>>> Fraction('3.141592653589793').limit_denominator(100)
Fraction(311, 99)
>>> Fraction(4321, 8765).limit_denominator(10000)
Fraction(4321, 8765)
```

Inherited from `fractions.Fraction`

`NonreducedFraction.reduce()`

Reduce nonreduced fraction:

```
>>> fraction = mathtools.NonreducedFraction(-6, 3)

>>> fraction.reduce()
Fraction(-2, 1)
```

Return fraction.

Special Methods

`NonreducedFraction.__abs__()`

Absolute value of nonreduced fraction:

```
>>> abs(mathtools.NonreducedFraction(-3, 3))
NonreducedFraction(3, 3)
```

Return nonreduced fraction.

`NonreducedFraction.__add__(expr)`

Add *expr* to *self*:

```
>>> mathtools.NonreducedFraction(3, 3) + 1
NonreducedFraction(6, 3)
```

Return nonreduced fraction.

`NonreducedFraction.__complex__()`
`complex(self) == complex(float(self), 0)`

Inherited from `numbers.Real`

`NonreducedFraction.__div__(expr)`
 Divide nonreduced fraction by *expr*:

```
>>> mathtools.NonreducedFraction(3, 3) / 1
NonreducedFraction(3, 3)
```

Return nonreduced fraction.

`NonreducedFraction.__divmod__(other)`
`divmod(self, other)`: The pair (`self // other`, `self % other`).

Sometimes this can be computed faster than the pair of operations.

Inherited from `numbers.Real`

`NonreducedFraction.__eq__(expr)`
 True when *expr* equals *self*:

```
>>> mathtools.NonreducedFraction(3, 3) == 1
True
```

Return boolean.

`NonreducedFraction.__float__()`
`float(self) = self.numerator / self.denominator`

It's important that this conversion use the integer's "true" division rather than casting one side to float before dividing so that ratios of huge integers convert without overflowing.

Inherited from `numbers.Rational`

`NonreducedFraction.__floordiv__(a, b)`
`a // b`

Inherited from `fractions.Fraction`

`NonreducedFraction.__ge__(expr)`
 True when nonreduced fraction is greater than or equal to *expr*:

```
>>> mathtools.NonreducedFraction(3, 3) >= 1
True
```

Return boolean.

`NonreducedFraction.__gt__(expr)`
 True when nonreduced fraction is greater than *expr*:

```
>>> mathtools.NonreducedFraction(3, 3) > 1
False
```

Return boolean.

`NonreducedFraction.__hash__()`
`hash(self)`

Tricky because values that are exactly representable as a float must have the same hash as that float.

Inherited from `fractions.Fraction`

`NonreducedFraction.__le__(expr)`

True when nonreduced fraction is less than or equal to *expr*:

```
>>> mathtools.NonreducedFraction(3, 3) <= 1
True
```

Return boolean.

`NonreducedFraction.__lt__(expr)`

True when nonreduced fraction is less than *expr*:

```
>>> mathtools.NonreducedFraction(3, 3) < 1
False
```

Return boolean.

`NonreducedFraction.__mod__(a, b)`

a % *b*

Inherited from `fractions.Fraction`

`NonreducedFraction.__mul__(expr)`

Multiply nonreduced fraction by *expr*:

```
>>> mathtools.NonreducedFraction(3, 3) * 3
NonreducedFraction(9, 3)
```

Return nonreduced fraction.

`NonreducedFraction.__ne__(expr)`

True when *expr* does not equal *self*:

```
>>> mathtools.NonreducedFraction(3, 3) != 'foo'
True
```

Return boolean.

`NonreducedFraction.__neg__()`

Negate nonreduced fraction:

```
>>> -mathtools.NonreducedFraction(3, 3)
NonreducedFraction(-3, 3)
```

Return nonreduced fraction.

`NonreducedFraction.__nonzero__(a)`

a != 0

Inherited from `fractions.Fraction`

`NonreducedFraction.__pos__(a)`

+*a*: Coerces a subclass instance to `Fraction`

Inherited from `fractions.Fraction`

`NonreducedFraction.__pow__(a, b)`

a ** *b*

If *b* is not an integer, the result will be a float or complex since roots are generally irrational. If *b* is an integer, the result will be rational.

Inherited from `fractions.Fraction`

`NonreducedFraction.__radd__(expr)`

Add nonreduced fraction to *expr*:

```
>>> 1 + mathtools.NonreducedFraction(3, 3)
NonreducedFraction(6, 3)
```

Return nonreduced fraction.

`NonreducedFraction.__rdiv__(expr)`
Divide *expr* by nonreduced fraction:

```
>>> 1 / mathtools.NonreducedFraction(3, 3)
NonreducedFraction(3, 3)
```

Return nonreduced fraction.

`NonreducedFraction.__rdivmod__(other)`
divmod(*other*, self): The pair (self // *other*, self % *other*).

Sometimes this can be computed faster than the pair of operations.

Inherited from `numbers.Real`

`NonreducedFraction.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`NonreducedFraction.__rfloordiv__(b, a)`
a // *b*

Inherited from `fractions.Fraction`

`NonreducedFraction.__rmod__(b, a)`
a % *b*

Inherited from `fractions.Fraction`

`NonreducedFraction.__rmul__(expr)`
Multiply *expr* by nonreduced fraction:

```
>>> 3 * mathtools.NonreducedFraction(3, 3)
NonreducedFraction(9, 3)
```

Return nonreduced fraction.

`NonreducedFraction.__rpow__(b, a)`
a ** *b*

Inherited from `fractions.Fraction`

`NonreducedFraction.__rsub__(expr)`
Subtract nonreduced fraction from *expr*:

```
>>> 1 - mathtools.NonreducedFraction(3, 3)
NonreducedFraction(0, 3)
```

Return nonreduced fraction.

`NonreducedFraction.__rtruediv__(b, a)`
a / *b*

Inherited from `fractions.Fraction`

`NonreducedFraction.__str__()`
String representation of nonreduced fraction:


```
>>> fraction = mathtools.NonreducedFraction(-6, 3)

>>> str(fraction)
'-6/3'
```

Return string.

`NonreducedFraction.__sub__(expr)`
Subtract *expr* from self:

```
>>> mathtools.NonreducedFraction(3, 3) - 2
NonreducedFraction(-3, 3)
```

Return nonreduced fraction.

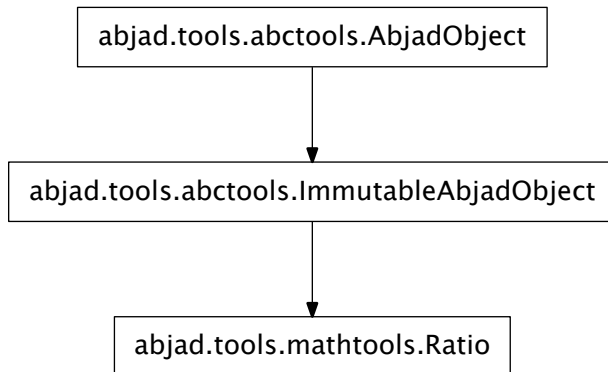
`NonreducedFraction.__truediv__(a, b)`
a / *b*

Inherited from `fractions.Fraction`

`NonreducedFraction.__trunc__(a)`
`trunc(a)`

Inherited from `fractions.Fraction`

`mathtools.Ratio`



class `mathtools.Ratio(*args, **kwargs)`
New in version 2.10. Ratio of two or more nonzero integers.

Initialize from two or more nonzero integers:

```
>>> from abjad.tools import mathtools

>>> mathtools.Ratio(1, 2, 1)
Ratio(1, 2, 1)
```

Or initialize from a tuple or list:

```
>>> ratio = mathtools.Ratio((1, 2, 1))
>>> ratio
Ratio(1, 2, 1)
```

Use a tuple to return ratio integers.

```
>>> tuple(ratio)
(1, 2, 1)
```

Ratios are immutable.

Read-only Properties

`Ratio.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`Ratio.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

`Ratio.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

Special Methods

`Ratio.__add__()`

`x.__add__(y) <==> x+y`

Inherited from `__builtin__.tuple`

`Ratio.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`Ratio.__eq__(other)`

`Ratio.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Ratio.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

`Ratio.__getslice__()`

`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.tuple`

`Ratio.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Ratio.__hash__() <==> hash(x)`

Inherited from `__builtin__.tuple`

`Ratio.__iter__() <==> iter(x)`

Inherited from `__builtin__.tuple`

`Ratio.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Ratio.__len__() <==> len(x)`

Inherited from `__builtin__.tuple`

`Ratio.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Ratio.__mul__()`

`x.__mul__(n) <==> x*n`

Inherited from `__builtin__.tuple`

`Ratio.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Ratio.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`Ratio.__rmul__()`

`x.__rmul__(n) <==> n*x`

Inherited from `__builtin__.tuple`

functions

`mathtools.are_relatively_prime`

`mathtools.are_relatively_prime(expr)`

New in version 2.5. True when *expr* is a sequence comprising zero or more numbers, all of which are relatively prime:

```
>>> from abjad.tools import mathtools

>>> mathtools.are_relatively_prime([13, 14, 15])
True
```

Otherwise false:

```
>>> mathtools.are_relatively_prime([13, 14, 15, 16])
False
```

Note that function returns true when *expr* is an empty sequence:

```
>>> mathtools.are_relatively_prime([])
True
```

Function returns false when *expr* is nonsensical type:

```
>>> mathtools.are_relatively_prime('foo')
False
```

Return boolean.

mathtools.arithmetic_mean

`mathtools.arithmetic_mean(sequence)`

New in version 1.1. Arithmetic means of *sequence* as an exact integer:

```
>>> from abjad.tools import mathtools

>>> mathtools.arithmetic_mean([1, 2, 2, 20, 30])
11
```

As a rational:

```
>>> mathtools.arithmetic_mean([1, 2, 20])
Fraction(23, 3)
```

As a float:

```
>>> mathtools.arithmetic_mean([2, 2, 20.0])
8.0
```

Return number. Changed in version 2.0: renamed `sequencetools.arithmetic_mean()` to `mathtools.arithmetic_mean()`.

mathtools.binomial_coefficient

`mathtools.binomial_coefficient(n, k)`

New in version 2.0. Binomial coefficient of *n* choose *k*:

```
>>> from abjad.tools import mathtools

>>> for k in range(8):
...     print k, '\t', mathtools.binomial_coefficient(8, k)
...
0  1
1  8
2  28
```

```

3  56
4  70
5  56
6  28
7   8

```

Return positive integer.

mathtools.cumulative_products

`mathtools.cumulative_products(sequence)`

Cumulative products of *sequence*:

```

>>> from abjad.tools import mathtools

>>> mathtools.cumulative_products([1, 2, 3, 4, 5, 6, 7, 8])
[1, 2, 6, 24, 120, 720, 5040, 40320]

>>> mathtools.cumulative_products([1, -2, 3, -4, 5, -6, 7, -8])
[1, -2, -6, 24, 120, -720, -5040, 40320]

```

Raise type error when *sequence* is neither list nor tuple.

Raise value error on empty *sequence*.

Return list. Changed in version 2.0: renamed `sequencetools.cumulative_products()` to `mathtools.cumulative_products()`.

mathtools.cumulative_signed_weights

`mathtools.cumulative_signed_weights(sequence)`

Cumulative signed weights of *sequence*:

```

>>> from abjad.tools import mathtools

>>> l = [1, -2, -3, 4, -5, -6, 7, -8, -9, 10]
>>> mathtools.cumulative_signed_weights(l)
[1, -3, -6, 10, -15, -21, 28, -36, -45, 55]

```

Raise type error when *sequence* is not a list.

For cumulative (unsigned) weights use `mathtools.cumulative_sums([abs(x) for x in l])`.

Return list. Changed in version 2.0: renamed `sequencetools.cumulative_weights_signed()` to `mathtools.cumulative_signed_weights()`.

mathtools.cumulative_sums

`mathtools.cumulative_sums(sequence)`

Cumulative sums of *sequence*:

```

>>> from abjad.tools import mathtools

>>> mathtools.cumulative_sums([1, 2, 3, 4, 5, 6, 7, 8])
[1, 3, 6, 10, 15, 21, 28, 36]

```

Raise type error when *sequence* is neither list nor tuple.

Raise value error on empty *sequence*.

Return list. Changed in version 2.0: renamed `sequencetools.cumulative_sums()` to `mathtools.cumulative_sums()`.

mathtools.cumulative_sums_zero

`mathtools.cumulative_sums_zero(sequence)`

Cumulative sums of *sequence* starting from 0:

```
>>> from abjad.tools import mathtools

>>> mathtools.cumulative_sums_zero([1, 2, 3, 4, 5, 6, 7, 8])
[0, 1, 3, 6, 10, 15, 21, 28, 36]
```

Return `[0]` on empty *sequence*:

```
>>> mathtools.cumulative_sums_zero([])
[0]
```

Return list. Changed in version 2.0: renamed `mathtools.cumulative_sums_zero()` to `mathtools.cumulative_sums_zero()`.

mathtools.cumulative_sums_zero_pairwise

`mathtools.cumulative_sums_zero_pairwise(sequence)`

List pairwise cumulative sums of *sequence* from 0:

```
>>> from abjad.tools import mathtools

>>> mathtools.cumulative_sums_zero_pairwise([1, 2, 3, 4, 5, 6])
[(0, 1), (1, 3), (3, 6), (6, 10), (10, 15), (15, 21)]
```

Return list of pairs. Changed in version 2.0: renamed `sequencetools.pairwise_cumulative_sums_zero()` to `mathtools.cumulative_sums_zero_pairwise()`.

mathtools.difference_series

`mathtools.difference_series(sequence)`

Difference series of *sequence*:

```
>>> from abjad.tools import mathtools

>>> mathtools.difference_series([1, 1, 2, 3, 5, 5, 6])
[0, 1, 1, 2, 0, 1]
```

Return list. Changed in version 2.0: renamed `sequencetools.difference_series()` to `mathtools.difference_series()`.

mathtools.divide_number_by_ratio

`mathtools.divide_number_by_ratio(number, ratio)`

Divide integer by *ratio*:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.divide_number_by_ratio(1, [1, 1, 3])
[Fraction(1, 5), Fraction(1, 5), Fraction(3, 5)]
```

Divide fraction by *ratio*:

```
>>> mathtools.divide_number_by_ratio(Fraction(1), [1, 1, 3])
[Fraction(1, 5), Fraction(1, 5), Fraction(3, 5)]
```

Divide float by ratio:

```
>>> mathtools.divide_number_by_ratio(1.0, [1, 1, 3])
[0.20000000000000001, 0.20000000000000001, 0.60000000000000009]
```

Raise type error on nonnumeric *number*.

Raise type error on noninteger in *ratio*.

Return list of fractions or list of floats. Changed in version 2.0: renamed `mathtools.divide_number_by_ratio()` to `mathtools.divide_number_by_ratio()`.

mathtools.divisors

`mathtools.divisors(n)`

Positive divisors of integer *n* in increasing order:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.divisors(84)
[1, 2, 3, 4, 6, 7, 12, 14, 21, 28, 42, 84]
```

```
>>> for x in range(10, 20):
...     print x, mathtools.divisors(x)
...
10 [1, 2, 5, 10]
11 [1, 11]
12 [1, 2, 3, 4, 6, 12]
13 [1, 13]
14 [1, 2, 7, 14]
15 [1, 3, 5, 15]
16 [1, 2, 4, 8, 16]
17 [1, 17]
18 [1, 2, 3, 6, 9, 18]
19 [1, 19]
```

Allow nonpositive *n*:

```
>>> mathtools.divisors(-27)
[1, 3, 9, 27]
```

Raise type error on noninteger *n*.

Raise not implemented error on 0.

Return list of positive integers.

mathtools.factors**mathtools.factors**(*n*)Integer factors of positive integer *n* in increasing order:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.factors(84)
[1, 2, 2, 3, 7]
```

```
>>> for n in range(10, 20):
...     print n, mathtools.factors(n)
...
10 [1, 2, 5]
11 [1, 11]
12 [1, 2, 2, 3]
13 [1, 13]
14 [1, 2, 7]
15 [1, 3, 5]
16 [1, 2, 2, 2, 2]
17 [1, 17]
18 [1, 2, 3, 3]
19 [1, 19]
```

Raise type error on noninteger *n*.Raise value error on nonpositive *n*.

Return list of one or more positive integers.

mathtools.get_shared_numeric_sign**mathtools.get_shared_numeric_sign**(*sequence*)Return 1 when all *sequence* elements are positive:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.get_shared_numeric_sign([1, 2, 3])
1
```

Return -1 when all *sequence* elements are negative:

```
>>> mathtools.get_shared_numeric_sign([-1, -2, -3])
-1
```

Return 0 on empty *sequence*:

```
>>> mathtools.get_shared_numeric_sign([])
0
```

Otherwise return none:

```
>>> mathtools.get_shared_numeric_sign([1, 2, -3]) is None
True
```

Return 1, -1, 0 or none. Changed in version 2.0: renamed `sequencetools.sign()` to `mathtools.get_shared_numeric_sign()`.

mathtools.greatest_common_divisor**mathtools.greatest_common_divisor** (**integers*)New in version 2.0. Greatest common divisor of *integers*:

```
>>> from abjad.tools import mathtools

>>> mathtools.greatest_common_divisor(84, -94, -144)
2
```

Allow nonpositive *integers*.Raise type error on noninteger *integers*.Raise not implemented error when 0 in *integers*.

Return positive integer.

mathtools.greatest_multiple_less_equal**mathtools.greatest_multiple_less_equal** (*m*, *n*)Greatest integer multiple of *m* less than or equal to *n*:

```
>>> from abjad.tools import mathtools

>>> mathtools.greatest_multiple_less_equal(10, 47)
40

>>> for m in range(1, 10):
...     print m, mathtools.greatest_multiple_less_equal(m, 47)
...
1 47
2 46
3 45
4 44
5 45
6 42
7 42
8 40
9 45

>>> for n in range(10, 100, 10):
...     print mathtools.greatest_multiple_less_equal(7, n), n
...
7 10
14 20
28 30
35 40
49 50
56 60
70 70
77 80
84 90
```

Raise type error on nonnumeric *m*.Raise type error on nonnumeric *n*.

Return nonnegative integer.

`mathtools.greatest_power_of_two_less_equal`

`mathtools.greatest_power_of_two_less_equal` (*n*, *i*=0)

Greatest integer power of two less than or equal to positive *n*:

```
>>> from abjad.tools import mathtools

>>> for n in range(10, 20):
...     print '\t%s\t%s' % (n, mathtools.greatest_power_of_two_less_equal(n))
...
10 8
11 8
12 8
13 8
14 8
15 8
16 16
17 16
18 16
19 16
```

Greatest-but-*i* integer power of 2 less than or equal to positive *n*:

```
>>> for n in range(10, 20):
...     print '\t%s\t%s' % (n, mathtools.greatest_power_of_two_less_equal(n, i=1))
...
10 4
11 4
12 4
13 4
14 4
15 4
16 8
17 8
18 8
19 8
```

Raise type error on nonnumeric *n*.

Raise value error on nonpositive *n*.

Return positive integer.

`mathtools.integer_equivalent_number_to_integer`

`mathtools.integer_equivalent_number_to_integer` (*number*)

New in version 2.0. Integer-equivalent *number* to integer:

```
>>> from abjad.tools import mathtools

>>> mathtools.integer_equivalent_number_to_integer(17.0)
17
```

Return noninteger-equivalent number unchanged:

```
>>> mathtools.integer_equivalent_number_to_integer(17.5)
17.5
```

Raise type error on nonnumber input.

Return number.

`mathtools.integer_to_base_k_tuple`

`mathtools.integer_to_base_k_tuple(n, k)`

New in version 2.0. Nonnegative integer n to base- k tuple:

```
>>> from abjad.tools import mathtools

>>> mathtools.integer_to_base_k_tuple(1066, 10)
(1, 0, 6, 6)
```

Return tuple of one or more positive integers.

`mathtools.integer_to_binary_string`

`mathtools.integer_to_binary_string(n)`

Positive integer n to binary string:

```
>>> from abjad.tools import mathtools

>>> mathtools.integer_to_binary_string(5)
'101'

>>> for n in range(1, 17):
...     print '\t%s\t%s' % (n, mathtools.integer_to_binary_string(n))
...
1  1
2  10
3  11
4  100
5  101
6  110
7  111
8  1000
9  1001
10 1010
11 1011
12 1100
13 1101
14 1110
15 1111
16 10000
```

Return string. Changed in version 2.0: renamed `mathtools.binary_string()` to `mathtools.integer_to_binary_string()`.

`mathtools.interpolate_cosine`

`mathtools.interpolate_cosine(y1, y2, mu)`

Cosine interpolate $y1$ and $y2$ with mu normalized $[0, 1]$:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.interpolate_cosine(0, 1, 0.5)
0.49999999999999994
```

Return float. Changed in version 2.0: renamed `interpolate.cosine()` to `mathtools.interpolate_cosine()`.

mathtools.interpolate_divide

`mathtools.interpolate_divide` (*total*, *start_fraction*, *stop_fraction*, *exp*='cosine')

Divide *total* into segments of sizes computed from interpolating between *start_fraction* and *stop_fraction*:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.interpolate_divide(10, 1, 1, exp=1)
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
>>> sum(mathtools.interpolate_divide(10, 1, 1, exp=1))
10.0
```

```
>>> mathtools.interpolate_divide(10, 5, 1)
[4.7986734489043181, 2.8792040693425909, 1.3263207210948171,
0.99580176065827419]
>>> sum(mathtools.interpolate_divide(10, 5, 1))
10.0
```

Set *exp*='cosine' for cosine interpolation.

Set *exp* to a numeric value for exponential interpolation with *exp* as the exponent.

Scale resulting segments so that their sum equals exactly *total*.

Return a list of floats. Changed in version 2.0: renamed `interpolate.divide()` to `mathtools.interpolate_divide()`.

mathtools.interpolate_divide_multiple

`mathtools.interpolate_divide_multiple` (*totals*, *key_values*, *exp*='cosine')

New in version 2.0. Interpolate *key_values* such that the sum of the resulting interpolated values equals the given *totals*:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.interpolate_divide_multiple([100, 50], [20, 10, 20])
[19.4487, 18.5201, 16.2270, 13.7156, 11.7488, 10.4879,
9.8515, 9.5130, 10.4213, 13.0736, 16.9918]
```

The operation is the same as `mathtools.interpolate_divide()`. But this function takes multiple *totals* and *key_values* at once.

Precondition: `len(totals) == len(key_values) - 1`.

Set *totals* equal to a list or tuple of the total sum of interpolated values.

Set *key_values* equal to a list or tuple of key values to interpolate.

Set *exp* to *cosine* for cosine interpolation.

Set *exp* to a number for exponential interpolation.

Returns a list of floats. Changed in version 2.0: renamed `interpolate.divide_multiple()` to `mathtools.interpolate_divide_multiple()`.

`mathtools.interpolate_exponential`

`mathtools.interpolate_exponential(y1, y2, mu, exp=1)`

Exponential interpolate *y1* and *y2* with *mu* normalized `[0, 1]`:

```
>>> from abjad.tools import mathtools

>>> mathtools.interpolate_exponential(0, 1, 0.5, 4)
0.0625
```

Set *exp* equal to the exponent of interpolation.

Return float. Changed in version 2.0: renamed `interpolate.exponential()` to `mathtools.interpolate_exponential()`.

`mathtools.interpolate_linear`

`mathtools.interpolate_linear(y1, y2, mu)`

Linear interpolate *y1* and *y2* with *mu* normalized `[0, 1]`:

```
>>> mathtools.interpolate_linear(0, 1, 0.5)
0.5
```

Return float. Changed in version 2.0: renamed `interpolate.linear()` to `mathtools.interpolate_linear()`.

`mathtools.interval_string_to_pair_and_indicators`

`mathtools.interval_string_to_pair_and_indicators(interval_string)`

New in version 1.0. Change *interval_string* to pair, boolean start indicator and boolean stop indicator:

```
>>> mathtools.interval_string_to_pair_and_indicators('[5, 8)')
((5, 8), False, True)
```

Parse square brackets as closed interval bounds.

Parse parentheses as open interval bounds.

Return triple.

`mathtools.is_assignable_integer`

`mathtools.is_assignable_integer(expr)`

New in version 2.0. True when *expr* is equivalent to an integer and can be written without recourse to ties:

```
>>> from abjad.tools import mathtools

>>> for n in range(0, 16 + 1):
...     print '%s\t%s' % (n, mathtools.is_assignable_integer(n))
...
0  False
1  True
```

```

2  True
3  True
4  True
5  False
6  True
7  True
8  True
9  False
10 False
11 False
12 True
13 False
14 True
15 True
16 True

```

Otherwise false.

Return boolean. Changed in version 2.0: renamed `mathtools.is_assignable()` to `mathtools.is_assignable_integer()`.

`mathtools.is_dotted_integer`

`mathtools.is_dotted_integer` (*expr*)

New in version 2.0. True when *expr* is equivalent to a positive integer and can be written with zero or more dots:

```

>>> from abjad.tools import mathtools

>>> for expr in range(16):
...     print '%s' % (expr, mathtools.is_dotted_integer(expr))
...
0      False
1      False
2      False
3      True
4      False
5      False
6      True
7      True
8      False
9      False
10     False
11     False
12     True
13     False
14     True
15     True

```

Otherwise false.

Return boolean.

Integer *n* qualifies as dotted when `abs(n)` is of the form `2**j * (2**k - 1)` with integers `0 <= j, 2 < k`.

`mathtools.is_integer_equivalent_expr`

`mathtools.is_integer_equivalent_expr(expr)`

New in version 2.5. True when *expr* is an integer-equivalent number:

```
>>> from abjad.tools import mathtools

>>> mathtools.is_integer_equivalent_expr(12.0)
True
```

True when *expr* evaluates to an integer:

```
>>> mathtools.is_integer_equivalent_expr('12')
True
```

Otherwise false:

```
>>> mathtools.is_integer_equivalent_expr('foo')
False
```

Return boolean.

`mathtools.is_integer_equivalent_number`

`mathtools.is_integer_equivalent_number(expr)`

New in version 2.0. True when *expr* is a number and *expr* is equivalent to an integer:

```
>>> from abjad.tools import mathtools

>>> mathtools.is_integer_equivalent_number(12.0)
True
```

Otherwise false:

```
>>> mathtools.is_integer_equivalent_number(Duration(1, 2))
False
```

Return boolean.

`mathtools.is_negative_integer`

`mathtools.is_negative_integer(expr)`

New in version 2.0. True when *expr* equals a negative integer:

```
>>> from abjad.tools import mathtools

>>> mathtools.is_negative_integer(-1)
True
```

Otherwise false:

```
>>> mathtools.is_negative_integer(0)
False

>>> mathtools.is_negative_integer(99)
False
```

Return boolean.

mathtools.is_nonnegative_integer

`mathtools.is_nonnegative_integer(expr)`

New in version 2.0. True when *expr* equals a nonnegative integer:

```
>>> from abjad.tools import mathtools

>>> mathtools.is_nonnegative_integer(99)
True

>>> mathtools.is_nonnegative_integer(0)
True
```

Otherwise false:

```
>>> mathtools.is_nonnegative_integer(-1)
False
```

Return boolean.

mathtools.is_nonnegative_integer_equivalent_number

`mathtools.is_nonnegative_integer_equivalent_number(expr)`

New in version 2.0. True when *expr* is a nonnegative integer-equivalent number. Otherwise false:

```
>>> from abjad.tools import mathtools

>>> mathtools.is_nonnegative_integer_equivalent_number(Duration(4, 2))
True
```

Return boolean.

mathtools.is_nonnegative_integer_power_of_two

`mathtools.is_nonnegative_integer_power_of_two(expr)`

True when *expr* is a nonnegative integer power of 2:

```
>>> from abjad.tools import mathtools

>>> for n in range(10):
...     print n, mathtools.is_nonnegative_integer_power_of_two(n)
...
0 True
1 True
2 True
3 False
4 True
5 False
6 False
7 False
8 True
9 False
```

Otherwise false.

Return boolean. Changed in version 2.0: renamed `mathtools.is_power_of_two()` to `mathtools.is_nonnegative_integer_power_of_two()`.

`mathtools.is_positive_integer`

`mathtools.is_positive_integer` (*expr*)

New in version 2.0. True when *expr* equals a positive integer:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.is_positive_integer(99)
True
```

Otherwise false:

```
>>> mathtools.is_positive_integer(0)
False
```

```
>>> mathtools.is_positive_integer(-1)
False
```

Return boolean.

`mathtools.is_positive_integer_equivalent_number`

`mathtools.is_positive_integer_equivalent_number` (*expr*)

New in version 2.0. True when *expr* is a positive integer-equivalent number. Otherwise false:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.is_positive_integer_equivalent_number(Duration(4, 2))
True
```

Return boolean.

`mathtools.is_positive_integer_power_of_two`

`mathtools.is_positive_integer_power_of_two` (*expr*)

True when *expr* is a positive integer power of 2:

```
>>> from abjad.tools import mathtools
```

```
>>> for n in range(10):
...     print n, mathtools.is_positive_integer_power_of_two(n)
...
0 False
1 True
2 True
3 False
4 True
5 False
6 False
7 False
8 True
9 False
```

Otherwise false.

Return boolean.

`mathtools.least_common_multiple`

`mathtools.least_common_multiple(*integers)`

Least common multiple of positive *integers*:

```
>>> from abjad.tools import mathtools

>>> mathtools.least_common_multiple(2, 4, 5, 10, 20)
20
```

Return positive integer.

`mathtools.least_multiple_greater_equal`

`mathtools.least_multiple_greater_equal(m, n)`

Return the least integer multiple of *m* greater than or equal to *n*.

```
>>> from abjad.tools import mathtools

>>> mathtools.least_multiple_greater_equal(10, 47)
50

>>> for m in range(1, 10):
...     print m, mathtools.least_multiple_greater_equal(m, 47)
...
1 47
2 48
3 48
4 48
5 50
6 48
7 49
8 48
9 54

>>> for n in range(10, 100, 10):
...     print mathtools.least_multiple_greater_equal(7, n), n
...
14 10
21 20
35 30
42 40
56 50
63 60
70 70
84 80
91 90
```

Return integer.

`mathtools.least_power_of_two_greater_equal`

`mathtools.least_power_of_two_greater_equal(n, i=0)`

Return least integer power of two greater than or equal to positive *n*:

```
>>> from abjad.tools import mathtools

>>> for n in range(10, 20):
...     print '\t%s\t%s' % (n, mathtools.least_power_of_two_greater_equal(n))
...
10 16
11 16
12 16
13 16
14 16
15 16
16 16
17 32
18 32
19 32
```

When $i = 1$, return the first integer power of 2 greater than the least integer power of 2 greater than or equal to n .

```
>>> for n in range(10, 20):
...     print '\t%s\t%s' % (n, mathtools.least_power_of_two_greater_equal(n, i=1))
...
10 32
11 32
12 32
13 32
14 32
15 32
16 32
17 64
18 64
19 64
```

When $i = 2$, return the second integer power of 2 greater than the least integer power of 2 greater than or equal to n , and, in general, return the i th integer power of 2 greater than the least integer power of 2 greater than or equal to n .

Raise type error on nonnumeric n .

Raise value error on nonpositive n .

Return integer.

mathtools.next_integer_partition

`mathtools.next_integer_partition(integer_partition)`

New in version 2.0. Next integer partition following *integer_partition* in descending lex order:

```
>>> mathtools.next_integer_partition((8, 3))
(8, 2, 1)

>>> mathtools.next_integer_partition((8, 2, 1))
(8, 1, 1, 1)

>>> mathtools.next_integer_partition((8, 1, 1, 1))
(7, 4)
```

Input *integer_partition* must be sequence of positive integers.

Return integer partition as tuple of positive integers.

`mathtools.partition_integer_by_ratio`

`mathtools.partition_integer_by_ratio(n, ratio)`

Partition positive integer-equivalent n by *ratio*:

```
>>> from abjad.tools import mathtools
```

```
>>> mathtools.partition_integer_by_ratio(10, [1, 2])
[3, 7]
```

Partition positive integer-equivalent n by *ratio* with negative parts:

```
>>> mathtools.partition_integer_by_ratio(10, [1, -2])
[3, -7]
```

Partition negative integer-equivalent n by *ratio*:

```
>>> mathtools.partition_integer_by_ratio(-10, [1, 2])
[-3, -7]
```

Partition negative integer-equivalent n by *ratio* with negative parts:

```
>>> mathtools.partition_integer_by_ratio(-10, [1, -2])
[-3, 7]
```

Return result with weight equal to absolute value of n .

Raise type error on noninteger n .

Return list of integers.

`mathtools.partition_integer_into_canonic_parts`

`mathtools.partition_integer_into_canonic_parts(n, big_endian=True)`

Partition integer n into big-endian or small-endian parts.

Return all parts positive on positive n :

```
>>> for n in range(1, 11):
...     print n, mathtools.partition_integer_into_canonic_parts(n)
...
1 (1,)
2 (2,)
3 (3,)
4 (4,)
5 (4, 1)
6 (6,)
7 (7,)
8 (8,)
9 (8, 1)
10 (8, 2)
```

Return all parts negative on negative n :

```
>>> for n in reversed(range(-20, -10)):
...     print n, mathtools.partition_integer_into_canonic_parts(n)
...
-11 (-8, -3)
-12 (-12,)
-13 (-12, -1)
-14 (-14,)
-15 (-15,)
-16 (-16,)
-17 (-16, -1)
-18 (-16, -2)
-19 (-16, -3)
-20 (-16, -4)
```

Return little-endian tuple:

```
>>> for n in range(11, 21):
...     print n, mathtools.partition_integer_into_canonic_parts(n, big_endian=False)
...
11 (3, 8)
12 (12,)
13 (1, 12)
14 (14,)
15 (15,)
16 (16,)
17 (1, 16)
18 (2, 16)
19 (3, 16)
20 (4, 16)
```

Return big-endian tuple $t = (t_0, \dots, t_j)$ such that

- $\text{sum}(t) == n$
- t_i can be written without recourse to ties, and
- $t_{(i+1)} < t_i$ for every t_i in t .

Raise type error on noninteger n .

Return tuple of one or more integers.

mathtools.partition_integer_into_halves

`mathtools.partition_integer_into_halves(n, bigger='left', even='allowed')`

Write positive integer n as the pair $t = (\text{left}, \text{right})$ such that $n == \text{left} + \text{right}$.

When n is odd the greater part of t corresponds to the value of *bigger*:

```
>>> from abjad.tools import mathtools

>>> mathtools.partition_integer_into_halves(7, bigger='left')
(4, 3)
>>> mathtools.partition_integer_into_halves(7, bigger='right')
(3, 4)
```

Likewise when n is even and `even = 'disallowed'`:

```
>>> mathtools.partition_integer_into_halves(8, bigger='left', even='disallowed')
(5, 3)
>>> mathtools.partition_integer_into_halves(8, bigger='right', even='disallowed')
(3, 5)
```

But when n is even and `even = 'allowed'` then `left == right` and *bigger* is ignored:

```
>>> mathtools.partition_integer_into_halves(8)
(4, 4)
>>> mathtools.partition_integer_into_halves(8, bigger='left')
(4, 4)
>>> mathtools.partition_integer_into_halves(8, bigger='right')
(4, 4)
```

When n is 0 return (0, 0):

```
>>> mathtools.partition_integer_into_halves(0)
(0, 0)
```

When n is 0 and `even = 'disallowed'` raise partition error.

Raise type error on noninteger n .

Raise value error on negative n .

Return pair of positive integers.

mathtools.partition_integer_into_units

`mathtools.partition_integer_into_units(n)`

Partition positive integer into units:

```
>>> from abjad.tools import mathtools

>>> mathtools.partition_integer_into_units(6)
[1, 1, 1, 1, 1, 1]
```

Partition negative integer into units:

```
>>> mathtools.partition_integer_into_units(-5)
[-1, -1, -1, -1, -1]
```

Partition 0 into units:

```
>>> mathtools.partition_integer_into_units(0)
[]
```

Return list of zero or more parts with absolute value equal to 1.

mathtools.remove_powers_of_two

`mathtools.remove_powers_of_two(n)`

Remove powers of 2 from the factors of positive integer n :

```
>>> from abjad.tools import mathtools
```

```
>>> for n in range(10, 100, 10):
...     print '\t%s\t%s' % (n, mathtools.remove_powers_of_two(n))
...
10 5
20 5
30 15
40 5
50 25
60 15
70 35
80 5
90 45
```

Raise type error on noninteger n .

Raise value error on nonpositive n .

Return positive integer.

mathtools.sign

`mathtools.sign(n)`

Return -1 on negative n :

```
>>> mathtools.sign(-96.2)
-1
```

Return 0 when n is 0:

```
>>> mathtools.sign(0)
0
```

Return 1 on positive n :

```
>>> mathtools.sign(Duration(9, 8))
1
```

Return -1, 0 or 1.

mathtools.weight

`mathtools.weight($sequence$)`

Sum of the absolute value of the elements in $sequence$:

```
>>> mathtools.weight([-1, -2, 3, 4, 5])
15
```

Return nonnegative integer. Changed in version 2.0: renamed `sequencetools.weight()` to `mathtools.weight()`.

mathtools.yield_all_compositions_of_integer

`mathtools.yield_all_compositions_of_integer(n)`

New in version 2.0. Yield all compositions of positive integer n in descending lex order:

```
>>> from abjad.tools import mathtools

>>> for integer_composition in mathtools.yield_all_compositions_of_integer(5):
...     integer_composition
...
(5,)
(4, 1)
(3, 2)
(3, 1, 1)
(2, 3)
(2, 2, 1)
(2, 1, 2)
(2, 1, 1, 1)
(1, 4)
(1, 3, 1)
(1, 2, 2)
(1, 2, 1, 1)
(1, 1, 3)
(1, 1, 2, 1)
(1, 1, 1, 2)
(1, 1, 1, 1, 1)
```

Integer compositions are ordered integer partitions.

Return generator of positive integer tuples of length at least 1. Changed in version 2.0: renamed `mathtools.integer_compositions()` to `mathtools.yield_all_compositions_of_integer()`.

`mathtools.yield_all_partitions_of_integer`

`mathtools.yield_all_partitions_of_integer(n)`

New in version 2.0. Yield all partitions of positive integer n in descending lex order:

```
>>> from abjad.tools import mathtools

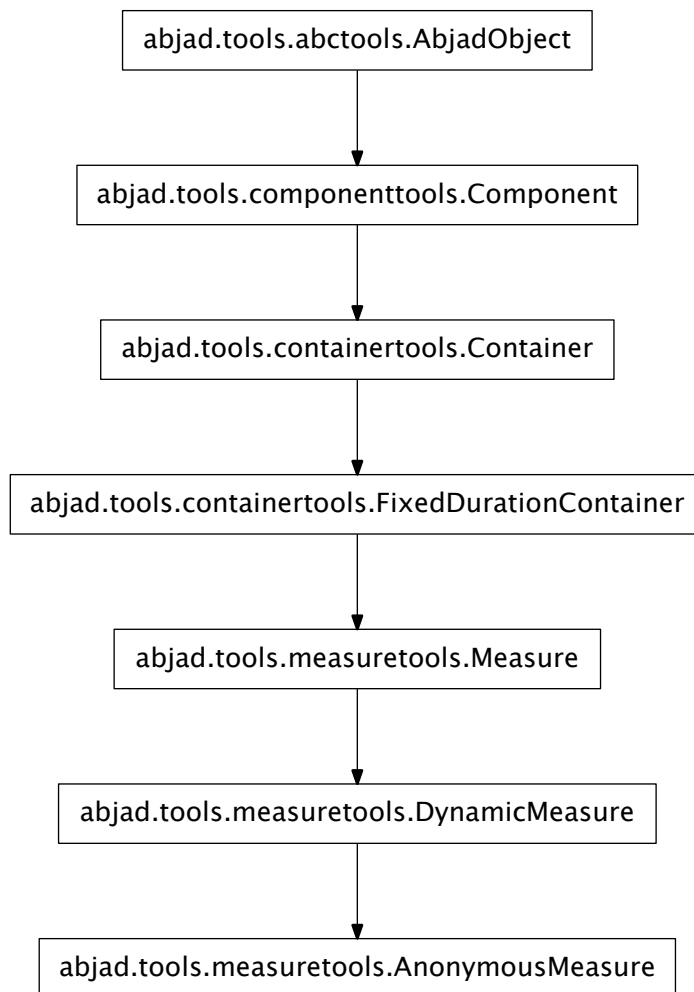
>>> for partition in mathtools.yield_all_partitions_of_integer(7):
...     partition
...
(7,)
(6, 1)
(5, 2)
(5, 1, 1)
(4, 3)
(4, 2, 1)
(4, 1, 1, 1)
(3, 3, 1)
(3, 2, 2)
(3, 2, 1, 1)
(3, 1, 1, 1, 1)
(2, 2, 2, 1)
(2, 2, 1, 1, 1)
(2, 1, 1, 1, 1, 1)
(1, 1, 1, 1, 1, 1, 1)
```

Return generator of positive integer tuples of length at least 1. Changed in version 2.0: renamed `mathtools.integer_partitions()` to `mathtools.yield_all_partitions_of_integer()`.

measuretools

concrete classes

measuretools.**AnonymousMeasure**



class measuretools.**AnonymousMeasure** (*music=None*, ***kwargs*)

New in version 1.1. Dynamic measure with no time signature:

```
>>> measure = measuretools.AnonymousMeasure("c'8 d'8 e'8 f'8")
```

```
>>> f(measure)
```

```
{
  \override Staff.TimeSignature #'stencil = ##f
  \time 1/2
```

```

        c'8
        d'8
        e'8
        f'8
        \revert Staff.TimeSignature #'stencil
    }

>>> notes = [Note("c'8"), Note("d'8")]
>>> measure.extend(notes)

>>> f(measure)
{
    \override Staff.TimeSignature #'stencil = ##f
    \time 3/4
    c'8
    d'8
    e'8
    f'8
    c'8
    d'8
    \revert Staff.TimeSignature #'stencil
}

```

Return anonymous measure.

Read-only Properties

`AnonymousMeasure.contents_duration`

Inherited from `containertools.Container`

`AnonymousMeasure.duration_in_seconds`

Inherited from `containertools.Container`

`AnonymousMeasure.is_binary`

True when measure time signature denominator is an integer power of 2:

```

>>> measure = Measure((5, 8), "c'8 d' e' f' g'")
>>> measure.is_binary
True

```

Otherwise false:

```

>>> measure = Measure((5, 9), "c'8 d' e' f' g'")
>>> measure.is_binary
False

```

Return boolean.

Inherited from `measuretools.Measure`

`AnonymousMeasure.is_full`

True when prolated duration equals time signature duration:

```

>>> measure = Measure((4, 8), "c'8 d'8 e'8 f'8")

>>> measure.is_full
True

```

Otherwise false.

Return boolean.

Inherited from `measuretools.Measure`

`AnonymousMeasure.is_misfilled`

New in version 2.9. True when measure is either underfull or overfull:

```
>>> measure = Measure((3, 4), "c' d' e' f'")

>>> measure
Measure(3/4, [c'4, d'4, e'4, f'4])

>>> measure.is_misfilled
True
```

Otherwise false:

```
>>> measure = Measure((3, 4), "c' d' e'")

>>> measure
Measure(3/4, [c'4, d'4, e'4])

>>> measure.is_misfilled
False
```

Return boolean.

Inherited from `measuretools.Measure`

`AnonymousMeasure.is_nonbinary`

True when measure time signature denominator is not an integer power of 2:

```
>>> measure = Measure((5, 9), "c'8 d' e' f' g'")
>>> measure.is_nonbinary
True
```

Otherwise false:

```
>>> measure = Measure((5, 8), "c'8 d' e' f' g'")
>>> measure.is_nonbinary
False
```

Return boolean.

Inherited from `measuretools.Measure`

`AnonymousMeasure.is_overfull`

New in version 1.1. True when prolated duration is greater than time signature duration:

```
>>> measure = Measure((3, 4), "c'4 d' e' f'")

>>> measure.is_overfull
True
```

Otherwise false.

Return boolean.

Inherited from `measuretools.Measure`

`AnonymousMeasure.is_underfull`

New in version 1.1. True when prolated duration is less than time signature duration:

```
>>> measure = Measure((3, 4), "c'4 d'")
```

```
>>> measure.is_underfull
True
```

Otherwise false.

Return boolean.

Inherited from `measuretools.Measure`

`AnonymousMeasure.leaves`

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

`AnonymousMeasure.lilypond_format`

Inherited from `measuretools.Measure`

`AnonymousMeasure.measure_number`

1-indexed measure number:

```
>>> staff = Staff()
>>> staff.append(Measure((3, 4), "c' d' e'"))
>>> staff.append(Measure((2, 4), "f' g'"))

>>> f(staff)
\new Staff {
    {
        \time 3/4
        c'4
        d'4
        e'4
    }
    {
        \time 2/4
        f'4
        g'4
    }
}

>>> staff[0].measure_number
1

>>> staff[1].measure_number
2
```

Return positive integer.

Inherited from `measuretools.Measure`

`AnonymousMeasure.multiplier`

Multiplier of measure time signature:

```
>>> measure = Measure((5, 12), "c'8 d' e' f' g'")
```

```
>>> measure.multiplier
Fraction(2, 3)
```

Return fraction.

Inherited from `measuretools.Measure`

`AnonymousMeasure.music`

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

`AnonymousMeasure.override`

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

`AnonymousMeasure.parent`

Inherited from `componenttools.Component`

`AnonymousMeasure.preprolated_duration`

Inherited from `measuretools.DynamicMeasure`

`AnonymousMeasure.prolated_duration`

Inherited from `componenttools.Component`

`AnonymousMeasure.prolation`

Inherited from `componenttools.Component`

`AnonymousMeasure.set`

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

`AnonymousMeasure.spanners`

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`AnonymousMeasure.start_offset`

Read-only start offset of component.

Inherited from `componenttools.Component`

`AnonymousMeasure.start_offset_in_seconds`

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

`AnonymousMeasure.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`AnonymousMeasure.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`AnonymousMeasure.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`AnonymousMeasure.target_duration`

New in version 2.9. Read-only target duration of measure always equal to duration of effective time signature.

Inherited from `measuretools.Measure`

Read/write Properties

`AnonymousMeasure.always_format_time_signature`

New in version 2.9. Read / write flag to indicate whether time signature should appear in LilyPond format even when not expected.

Set to true when necessary to print the same signature repeatedly.

Default to false.

Return boolean.

Inherited from `measuretools.Measure`

`AnonymousMeasure.automatically_adjust_time_signature`

New in version 2.9. Read / write flag to indicate whether time signature should update automatically following contents-changing operations:

```
>>> measure = Measure((3, 4), "c' d' e'")

>>> measure
Measure(3/4, [c'4, d'4, e'4])

>>> measure.automatically_adjust_time_signature = True
>>> measure.append('r')

>>> measure
Measure(4/4, [c'4, d'4, e'4, r4])
```

Default to false.

Return boolean.

Inherited from `measuretools.Measure`

`AnonymousMeasure.denominator`

Get explicit denominator of dynamic measure:

```
>>> measure = measuretools.DynamicMeasure("c'8 d'8 e'8 f'8")

>>> measure.denominator is None
True
```

Set explicit denominator of dynamic measure:

```
>>> measure.denominator = 8

>>> f(measure)
{
  \time 4/8
  c'8
  d'8
```

```

        e'8
        f'8
    }

```

Set positive integer or none.

Inherited from `measuretools.DynamicMeasure`

`AnonymousMeasure.is_parallel`

Get parallel container:

```

>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')] )

>>> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

>>> container.is_parallel
False

```

Return boolean.

Set parallel container:

```

>>> container.is_parallel = True

>>> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>

```

Return none.

Inherited from `containertools.Container`

`AnonymousMeasure.suppress_meter`

Get meter suppression indicator:

```

>>> measure = measuretools.DynamicMeasure("c'8 d'8 e'8 f'8")

>>> f(measure)
{
    \time 1/2
    c'8
    d'8
    e'8

```

```

        f'8
    }

```

```

>>> measure.suppress_meter
False

```

Set meter suppression indicator:

```

>>> measure.suppress_meter = True

```

```

>>> measure.suppress_meter
True

```

```

>>> f(measure)
{
    c'8
    d'8
    e'8
    f'8
}

```

Set boolean.

Inherited from `measuretools.DynamicMeasure`

Methods

`AnonymousMeasure.append(component)`

Append *component* to container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

```

```

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

```

```

>>> container.append(Note("f'8"))

```

```

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}

```

Return none.

Inherited from `containertools.Container`

`AnonymousMeasure.extend(expr)`

Extend dynamic measure:

```

>>> measure = measuretools.DynamicMeasure("c'8 d'8 e'8")

```



```
>>> f(measure)
{
    \time 3/8
    c'8
    d'8
    e'8
}

>>> measure.extend([Note("f'8"), Note("g'8")])

>>> f(measure)
{
    \time 5/8
    c'8
    d'8
    e'8
    f'8
    g'8
}
```

Return none.

Inherited from `measuretools.DynamicMeasure`

`AnonymousMeasure.index(component)`

Index *component* in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

`AnonymousMeasure.insert(i, component)`

Insert *component* in container at index *i*:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
}
```

```

        e'8 ]
    }

```

Return none.

Inherited from `containertools.Container`

`AnonymousMeasure.pop` (*i=-1*)

Pop component at index *i* from container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}

```

Return component.

Inherited from `containertools.Container`

`AnonymousMeasure.remove` (*component*)

Remove *component* from container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}

```

Return none.

Inherited from `containertools.Container`

Special Methods

`AnonymousMeasure.__add__(arg)`

Add two measures together in-score or outside-of-score. Wrapper around `measuretools.fuse_measures`.

Inherited from `measuretools.Measure`

`AnonymousMeasure.__contains__(expr)`

True if `expr` is in container, otherwise False.

Inherited from `containertools.Container`

`AnonymousMeasure.__delitem__(i)`

Container item deletion with optional time signature adjustment.

Inherited from `measuretools.Measure`

`AnonymousMeasure.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__getitem__(i)`

Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`AnonymousMeasure.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`AnonymousMeasure.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

`AnonymousMeasure.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`AnonymousMeasure.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__mul__(n)`

Inherited from `componenttools.Component`

`AnonymousMeasure.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`AnonymousMeasure.__repr__()`

String form of measure with parentheses for interpreter display.

Inherited from `measuretools.Measure`

`AnonymousMeasure.__rmul__(n)`

Inherited from `componenttools.Component`

`AnonymousMeasure.__setitem__(i, expr)`

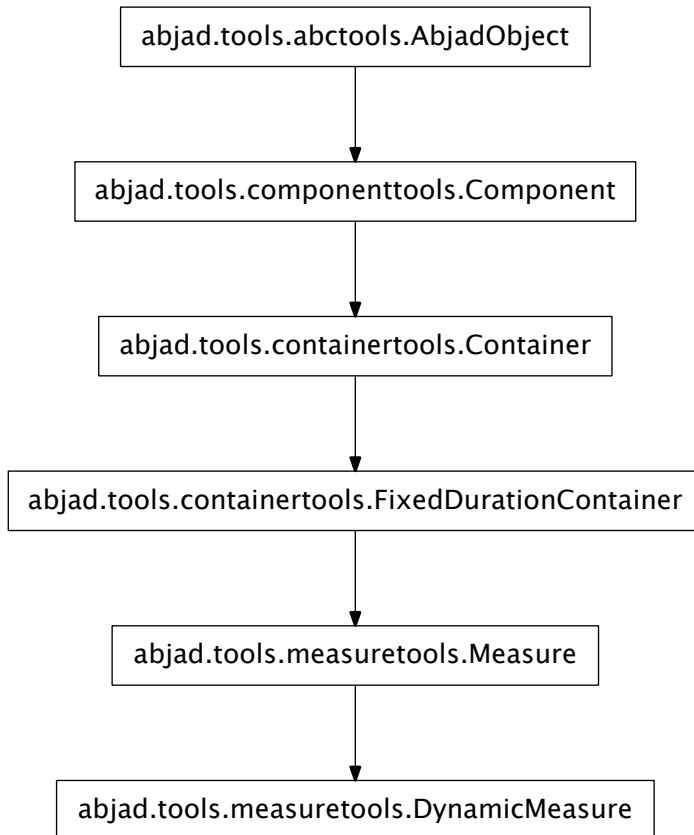
Container setitem logic with optional time signature adjustment. Changed in version 2.9: Measure setitem logic now adjusts time signature automatically when `adjust_time_signature_automatically` is true.

Inherited from `measuretools.Measure`

`AnonymousMeasure.__str__()`

String form of measure with pipes for single string display.

Inherited from `measuretools.Measure`

measuretools.DynamicMeasure

class `measuretools.DynamicMeasure` (*music=None*, ***kwargs*)

New in version 1.1. Measure sets meter dynamically to exactly equal contents duration:

```
>>> measure = measuretools.DynamicMeasure("c'8 d'8 e'8")
```

```
>>> measure
DynamicMeasure(3/8, [c'8, d'8, e'8])
```

```
>>> f(measure)
{
    \time 3/8
    c'8
    d'8
    e'8
}
```

Return dynamic measure.

Read-only Properties

`DynamicMeasure.contents_duration`

Inherited from `containertools.Container`

`DynamicMeasure.duration_in_seconds`

Inherited from `containertools.Container`

`DynamicMeasure.is_binary`

True when measure time signature denominator is an integer power of 2:

```
>>> measure = Measure((5, 8), "c'8 d' e' f' g'")
>>> measure.is_binary
True
```

Otherwise false:

```
>>> measure = Measure((5, 9), "c'8 d' e' f' g'")
>>> measure.is_binary
False
```

Return boolean.

Inherited from `measuretools.Measure`

`DynamicMeasure.is_full`

True when prolated duration equals time signature duration:

```
>>> measure = Measure((4, 8), "c'8 d'8 e'8 f'8")

>>> measure.is_full
True
```

Otherwise false.

Return boolean.

Inherited from `measuretools.Measure`

`DynamicMeasure.is_misfilled`

New in version 2.9. True when measure is either underfull or overfull:

```
>>> measure = Measure((3, 4), "c' d' e' f'")

>>> measure
Measure(3/4, [c'4, d'4, e'4, f'4])

>>> measure.is_misfilled
True
```

Otherwise false:

```
>>> measure = Measure((3, 4), "c' d' e'")

>>> measure
Measure(3/4, [c'4, d'4, e'4])

>>> measure.is_misfilled
False
```

Return boolean.

Inherited from `measuretools.Measure`

`DynamicMeasure.is_nonbinary`

True when measure time signature denominator is not an integer power of 2:

```
>>> measure = Measure((5, 9), "c'8 d' e' f' g'")
>>> measure.is_nonbinary
True
```

Otherwise false:

```
>>> measure = Measure((5, 8), "c'8 d' e' f' g'")
>>> measure.is_nonbinary
False
```

Return boolean.

Inherited from `measuretools.Measure`

`DynamicMeasure.is_overfull`

New in version 1.1. True when prolated duration is greater than time signature duration:

```
>>> measure = Measure((3, 4), "c'4 d' e' f'")

>>> measure.is_overfull
True
```

Otherwise false.

Return boolean.

Inherited from `measuretools.Measure`

`DynamicMeasure.is_underfull`

New in version 1.1. True when prolated duration is less than time signature duration:

```
>>> measure = Measure((3, 4), "c'4 d'")

>>> measure.is_underfull
True
```

Otherwise false.

Return boolean.

Inherited from `measuretools.Measure`

`DynamicMeasure.leaves`

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

`DynamicMeasure.lilypond_format`

Inherited from `measuretools.Measure`

`DynamicMeasure.measure_number`

1-indexed measure number:

```
>>> staff = Staff()
>>> staff.append(Measure((3, 4), "c' d' e'"))
>>> staff.append(Measure((2, 4), "f' g'"))
```

```
>>> f(staff)
\new Staff {
  {
    \time 3/4
    c'4
    d'4
    e'4
  }
  {
    \time 2/4
    f'4
    g'4
  }
}

>>> staff[0].measure_number
1

>>> staff[1].measure_number
2
```

Return positive integer.

Inherited from `measuretools.Measure`

`DynamicMeasure.multiplier`

Multiplier of measure time signature:

```
>>> measure = Measure((5, 12), "c'8 d' e' f' g'")

>>> measure.multiplier
Fraction(2, 3)
```

Return fraction.

Inherited from `measuretools.Measure`

`DynamicMeasure.music`

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

`DynamicMeasure.override`

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

`DynamicMeasure.parent`

Inherited from `componenttools.Component`

`DynamicMeasure.preprolated_duration`

`DynamicMeasure.prolated_duration`

Inherited from `componenttools.Component`

`DynamicMeasure.prolation`

Inherited from `componenttools.Component`

`DynamicMeasure.set`

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

`DynamicMeasure.spanners`

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`DynamicMeasure.start_offset`

Read-only start offset of component.

Inherited from `componenttools.Component`

`DynamicMeasure.start_offset_in_seconds`

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

`DynamicMeasure.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`DynamicMeasure.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`DynamicMeasure.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`DynamicMeasure.target_duration`

New in version 2.9. Read-only target duration of measure always equal to duration of effective time signature.

Inherited from `measuretools.Measure`

Read/write Properties

`DynamicMeasure.always_format_time_signature`

New in version 2.9. Read / write flag to indicate whether time signature should appear in LilyPond format even when not expected.

Set to true when necessary to print the same signature repeatedly.

Default to false.

Return boolean.

Inherited from `measuretools.Measure`

`DynamicMeasure.automatically_adjust_time_signature`

New in version 2.9. Read / write flag to indicate whether time signature should update automatically following contents-changing operations:

```
>>> measure = Measure((3, 4), "c' d' e' ")
```

```
>>> measure
Measure(3/4, [c'4, d'4, e'4])

>>> measure.automatically_adjust_time_signature = True
>>> measure.append('r')

>>> measure
Measure(4/4, [c'4, d'4, e'4, r4])
```

Default to false.

Return boolean.

Inherited from `measuretools.Measure`

`DynamicMeasure.denominator`

Get explicit denominator of dynamic measure:

```
>>> measure = measuretools.DynamicMeasure("c'8 d'8 e'8 f'8")

>>> measure.denominator is None
True
```

Set explicit denominator of dynamic measure:

```
>>> measure.denominator = 8

>>> f(measure)
{
    \time 4/8
    c'8
    d'8
    e'8
    f'8
}
```

Set positive integer or none.

`DynamicMeasure.is_parallel`

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

>>> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

>>> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True

>>> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Inherited from `containertools.Container`

`DynamicMeasure.suppress_meter`

Get meter suppression indicator:

```
>>> measure = measuretools.DynamicMeasure("c'8 d'8 e'8 f'8")

>>> f(measure)
{
    \time 1/2
    c'8
    d'8
    e'8
    f'8
}
```

```
>>> measure.suppress_meter
False
```

Set meter suppression indicator:

```
>>> measure.suppress_meter = True

>>> measure.suppress_meter
True

>>> f(measure)
{
    c'8
    d'8
    e'8
    f'8
}
```

Set boolean.

Methods

`DynamicMeasure.append(component)`

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
  f'8
}
```

Return none.

Inherited from `containertools.Container`

`DynamicMeasure.extend(expr)`

Extend dynamic measure:

```
>>> measure = measuretools.DynamicMeasure("c'8 d'8 e'8")

>>> f(measure)
{
  \time 3/8
  c'8
  d'8
  e'8
}

>>> measure.extend([Note("f'8"), Note("g'8")])

>>> f(measure)
{
  \time 5/8
  c'8
  d'8
  e'8
  f'8
  g'8
}
```

Return none.

`DynamicMeasure.index(component)`

Index *component* in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

`DynamicMeasure.insert(i, component)`

Insert *component* in container at index *i*:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Inherited from `containertools.Container`

`DynamicMeasure.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

`DynamicMeasure.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
  c'8 [
  d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

`DynamicMeasure.__add__(arg)`

Add two measures together in-score or outside-of-score. Wrapper around `measuretools.fuse_measures`.

Inherited from `measuretools.Measure`

`DynamicMeasure.__contains__(expr)`

True if `expr` is in container, otherwise False.

Inherited from `containertools.Container`

`DynamicMeasure.__delitem__(i)`

Container item deletion with optional time signature adjustment.

Inherited from `measuretools.Measure`

`DynamicMeasure.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DynamicMeasure.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DynamicMeasure.__getitem__(i)`

Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`DynamicMeasure.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DynamicMeasure.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`DynamicMeasure.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

`DynamicMeasure.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DynamicMeasure.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`DynamicMeasure.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DynamicMeasure.__mul__(n)`

Inherited from `componenttools.Component`

`DynamicMeasure.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DynamicMeasure.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`DynamicMeasure.__repr__()`

String form of measure with parentheses for interpreter display.

Inherited from `measuretools.Measure`

`DynamicMeasure.__rmul__(n)`

Inherited from `componenttools.Component`

`DynamicMeasure.__setitem__(i, expr)`

Container setitem logic with optional time signature adjustment. Changed in version 2.9: Measure setitem logic now adjusts time signature automatically when `adjust_time_signature_automatically` is true.

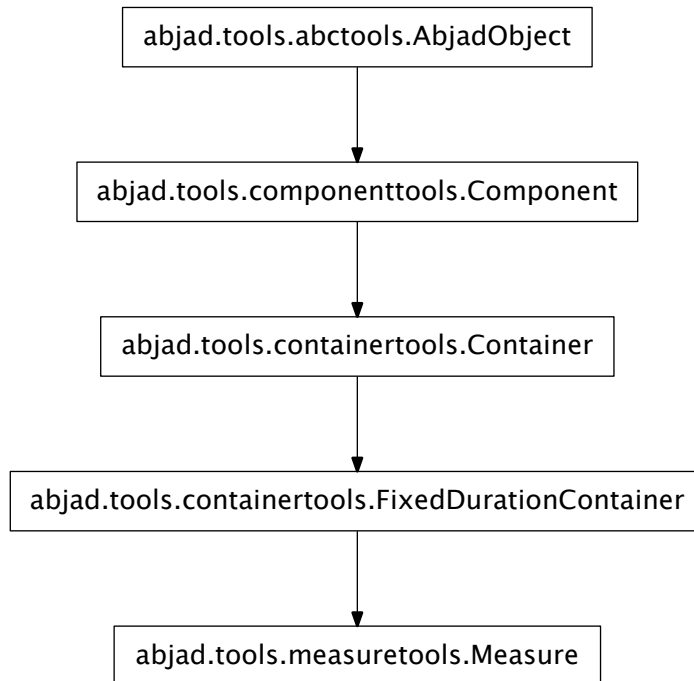
Inherited from `measuretools.Measure`

`DynamicMeasure.__str__()`

String form of measure with pipes for single string display.

Inherited from `measuretools.Measure`

`measuretools.Measure`



class `measuretools.Measure` (*meter*, *music=None*, ***kwargs*)

New in version 1.1.Changed in version 2.9: measure now inherits from fixed-duration container. Abjad model of a measure:

```
>>> measure = Measure((4, 8), "c'8 d' e' f'")
```

```
>>> measure
Measure(4/8, [c'8, d'8, e'8, f'8])
```

```
>>> f(measure)
{
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}
```

Return measure object.

Read-only Properties

Measure.contents_duration

Inherited from `containertools.Container`

Measure.duration_in_seconds

Inherited from `containertools.Container`

Measure.is_binary

True when measure time signature denominator is an integer power of 2:

```
>>> measure = Measure((5, 8), "c'8 d' e' f' g'")
>>> measure.is_binary
True
```

Otherwise false:

```
>>> measure = Measure((5, 9), "c'8 d' e' f' g'")
>>> measure.is_binary
False
```

Return boolean.

Measure.is_full

True when prolated duration equals time signature duration:

```
>>> measure = Measure((4, 8), "c'8 d'8 e'8 f'8")

>>> measure.is_full
True
```

Otherwise false.

Return boolean.

Measure.is_misfilled

New in version 2.9. True when measure is either underfull or overfull:

```
>>> measure = Measure((3, 4), "c' d' e' f'")

>>> measure
Measure(3/4, [c'4, d'4, e'4, f'4])

>>> measure.is_misfilled
True
```

Otherwise false:

```
>>> measure = Measure((3, 4), "c' d' e'")

>>> measure
Measure(3/4, [c'4, d'4, e'4])

>>> measure.is_misfilled
False
```

Return boolean.

Measure.is_nonbinary

True when measure time signature denominator is not an integer power of 2:

```
>>> measure = Measure((5, 9), "c'8 d' e' f' g'")
>>> measure.is_nonbinary
True
```

Otherwise false:

```
>>> measure = Measure((5, 8), "c'8 d' e' f' g'")
>>> measure.is_nonbinary
False
```

Return boolean.

Measure.is_overfull

New in version 1.1. True when prolated duration is greater than time signature duration:

```
>>> measure = Measure((3, 4), "c'4 d' e' f'")

>>> measure.is_overfull
True
```

Otherwise false.

Return boolean.

Measure.is_underfull

New in version 1.1. True when prolated duration is less than time signature duration:

```
>>> measure = Measure((3, 4), "c'4 d'")

>>> measure.is_underfull
True
```

Otherwise false.

Return boolean.

Measure.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

Measure.lilypond_format

Measure.measure_number

1-indexed measure number:

```
>>> staff = Staff()
>>> staff.append(Measure((3, 4), "c' d' e'"))
>>> staff.append(Measure((2, 4), "f' g'"))

>>> f(staff)
\new Staff {
  {
    \time 3/4
    c'4
    d'4
    e'4
  }
  {
    \time 2/4
    f'4
  }
}
```

```

        g' 4
    }
}

>>> staff[0].measure_number
1

>>> staff[1].measure_number
2

```

Return positive integer.

Measure.**multiplier**

Multiplier of measure time signature:

```

>>> measure = Measure((5, 12), "c'8 d' e' f' g'")

>>> measure.multiplier
Fraction(2, 3)

```

Return fraction.

Measure.**music**

Read-only tuple of components in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))

```

Return tuple or zero or more components.

Inherited from `containertools.Container`

Measure.**override**

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Measure.**parent**

Inherited from `componenttools.Component`

Measure.**preprolated_duration**

Preprolated duration of measure:

```

>>> measure = Measure((5, 12), "c'8 d' e' f' g'")

>>> measure.preprolated_duration
Duration(5, 12)

```

Equal measure contents duration times time signature multiplier.

Return duration.

Measure.**prolated_duration**

Inherited from `componenttools.Component`

Measure.**prolation**

Inherited from `componenttools.Component`

Measure.**set**

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Measure.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Measure.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

Measure.start_offset_in_seconds

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

Measure.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

Measure.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Measure.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Measure.target_duration

New in version 2.9. Read-only target duration of measure always equal to duration of effective time signature.

Read/write Properties

Measure.always_format_time_signature

New in version 2.9. Read / write flag to indicate whether time signature should appear in LilyPond format even when not expected.

Set to true when necessary to print the same signature repeatedly.

Default to false.

Return boolean.

Measure.automatically_adjust_time_signature

New in version 2.9. Read / write flag to indicate whether time signature should update automatically following contents-changing operations:

```
>>> measure = Measure((3, 4), "c' d' e' ")
>>> measure
Measure(3/4, [c'4, d'4, e'4])
>>> measure.automatically_adjust_time_signature = True
>>> measure.append('r')
>>> measure
Measure(4/4, [c'4, d'4, e'4, r4])
```

Default to false.

Return boolean.

Measure.**is_parallel**

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')] )

>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}

>>> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True

>>> f(container)
<<
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
>>
```

Return none.

Inherited from `containertools.Container`

Methods

Measure.**append**(*component*)

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
}
```

```
>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

`Measure.extend(expr)`

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: *expr* may now be a LilyPond input string. Inherited from `containertools.Container`

`Measure.index(component)`

Index *component* in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

`Measure.insert(i, component)`

Insert *component* in container at index *i*:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}

```

Return none.

Inherited from `containertools.Container`

Measure.**pop** (*i=-1*)

Pop component at index *i* from container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}

```

Return component.

Inherited from `containertools.Container`

Measure.**remove** (*component*)

Remove *component* from container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8

```

```

        e'8 ]
    }

    >>> note = container[-1]
    >>> note
    Note("e'8")

    >>> container.remove(note)

    >>> f(container)
    {
        c'8 [
        d'8 ]
    }

```

Return none.

Inherited from `containertools.Container`

Special Methods

`Measure.__add__(arg)`

Add two measures together in-score or outside-of-score. Wrapper around `measuretools.fuse_measures`.

`Measure.__contains__(expr)`

True if `expr` is in container, otherwise False.

Inherited from `containertools.Container`

`Measure.__delitem__(i)`

Container item deletion with optional time signature adjustment.

`Measure.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Measure.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Measure.__getitem__(i)`

Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`Measure.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Measure.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

Measure.__**imul**__(total)

Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

Measure.__**le**__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Measure.__**len**__()

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

Measure.__**lt**__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Measure.__**mul**__(n)

Inherited from `componenttools.Component`

Measure.__**ne**__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Measure.__**radd**__(expr)

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

Measure.__**repr**__()

String form of measure with parentheses for interpreter display.

Measure.__**rmul**__(n)

Inherited from `componenttools.Component`

Measure.__**setitem**__(i, expr)

Container setitem logic with optional time signature adjustment. Changed in version 2.9: Measure setitem logic now adjusts time signature automatically when `adjust_time_signature_automatically` is true.

Measure.__**str**__()

String form of measure with pipes for single string display.

functions

`measuretools.all_are_measures`

`measuretools.all_are_measures`(expr)

New in version 2.6. True when `expr` is a sequence of Abjad measures:

```
>>> measures = 3 * Measure((3, 4), "c'4 d'4 e'4")
```

```
>>> for measure in measures:
...     measure
...
Measure(3/4, [c'4, d'4, e'4])
Measure(3/4, [c'4, d'4, e'4])
Measure(3/4, [c'4, d'4, e'4])

>>> measuretools.all_are_measures(measures)
True
```

True when *expr* is an empty sequence:

```
>>> measuretools.all_are_measures([])
True
```

Otherwise false:

```
>>> measuretools.all_are_measures('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

measuretools.append_spacer_skip_to_underfull_measure

`measuretools.append_spacer_skip_to_underfull_measure(rigid_measure)`

New in version 1.1. Append spacer skip to underfull *measure*:

```
>>> measure = Measure((4, 12), "c'8 d'8 e'8 f'8")
>>> contexttools.detach_time_signature_marks_attached_to_component(measure)
(TimeSignatureMark((4, 12)),)
>>> contexttools.TimeSignatureMark((5, 12))(measure)
TimeSignatureMark((5, 12))(|5/12, c'8, d'8, e'8, f'8|)
>>> measure.is_underfull
True

>>> measuretools.append_spacer_skip_to_underfull_measure(measure)
Measure(5/12, [c'8, d'8, e'8, f'8, s1 * 1/8])

>>> f(measure)
{
    \time 5/12
    \scaleDurations #'(2 . 3) {
        c'8
        d'8
        e'8
        f'8
        s1 * 1/8
    }
}
```

Append nothing to nonunderfull *measure*.

Return *measure*.

measuretools.append_spacer_skips_to_underfull_measures_in_expr

`measuretools.append_spacer_skips_to_underfull_measures_in_expr(expr)`

New in version 1.1. Append spacer skips to underfull measures in *expr*:

```
>>> staff = Staff(Measure((3, 8), "c'8 d'8 e'8") * 3)
>>> contexttools.detach_time_signature_marks_attached_to_component(staff[1])
(TimeSignatureMark((3, 8)),)
>>> contexttools.TimeSignatureMark((4, 8))(staff[1])
TimeSignatureMark((4, 8))(|4/8, c'8, d'8, e'8|)
>>> contexttools.detach_time_signature_marks_attached_to_component(staff[2])
(TimeSignatureMark((3, 8)),)
>>> contexttools.TimeSignatureMark((5, 8))(staff[2])
TimeSignatureMark((5, 8))(|5/8, c'8, d'8, e'8|)
>>> staff[1].is_underfull
True
>>> staff[2].is_underfull
True

>>> measuretools.append_spacer_skips_to_underfull_measures_in_expr(staff)
[Measure(4/8, [c'8, d'8, e'8, s1 * 1/8]), Measure(5/8, [c'8, d'8, e'8, s1 * 1/4])]

>>> f(staff)
\new Staff {
  {
    \time 3/8
    c'8
    d'8
    e'8
  }
  {
    \time 4/8
    c'8
    d'8
    e'8
    s1 * 1/8
  }
  {
    \time 5/8
    c'8
    d'8
    e'8
    s1 * 1/4
  }
}
```

Return measures treated. Changed in version 2.0: renamed `measuretools.remedy_underfull_measures()` to `measuretools.append_spacer_skips_to_underfull_measures_in_expr()`.

measuretools.apply_full_measure_tuplets_to_contents_of_measures_in_expr

`measuretools.apply_full_measure_tuplets_to_contents_of_measures_in_expr(expr)`

New in version 2.0. Apply full-measure tuplets to contents of measures in *expr*:

```
>>> staff = Staff([Measure((2, 8), "c'8 d'8"), Measure((3, 8), "e'8 f'8 g'8")])
```

```
>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 3/8
    e'8
    f'8
    g'8
  }
}

>>> measuretools.apply_full_measure_tuplets_to_contents_of_measures_in_expr(staff)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    {
      c'8
      d'8
    }
  }
  {
    \time 3/8
    {
      e'8
      f'8
      g'8
    }
  }
}
```

Return none.

measuretools.comment_measures_in_container_with_measure_numbers

measuretools.comment_measures_in_container_with_measure_numbers(*container*)

New in version 1.1. Comment measures in *container* with measure numbers:

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> measuretools.comment_measures_in_container_with_measure_numbers(staff)

>>> f(staff)
\new Staff {
  % start measure 1
  {
    \time 2/8
    c'8
    d'8
  }
  % stop measure 1
}
```

```

% start measure 2
{
    e'8
    f'8
}
% stop measure 2
% start measure 3
{
    g'8
    a'8
}
% stop measure 3
}

```

Changed in version 2.0: renamed `label.measure_numbers()` to `measuretools.comment_measures_in_container_with_measure_numbers()`.

`measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets`

`measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets`(*expr*,
supplement)

New in version 2.0. Extend measures in *expr* with *supplement* and apply full-measure tuplets to contents of measures:

```

>>> staff = Staff([Measure((2, 8), "c'8 d'8"), Measure((3, 8), "e'8 f'8 g'8")])

>>> f(staff)
\new Staff {
{
    \time 2/8
    c'8
    d'8
}
{
    \time 3/8
    e'8
    f'8
    g'8
}
}

>>> supplement = [Rest((1, 16))]
>>> measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets(
... staff, supplement)

>>> f(staff)
\new Staff {
{
    \time 2/8
    \times 4/5 {
        c'8
        d'8
        r16
    }
}
{

```

```

        \time 3/8
        \fraction \times 6/7 {
            e'8
            f'8
            g'8
            r16
        }
    }
}

```

Return none.

measuretools.fill_measures_in_expr_with_full_measure_spacer_skips

`measuretools.fill_measures_in_expr_with_full_measure_spacer_skips`(*expr*, *iterc-trl=None*)

New in version 1.1. Fill measures in *expr* with full-measure spacer skips.

measuretools.fill_measures_in_expr_with_minimal_number_of_notes

`measuretools.fill_measures_in_expr_with_minimal_number_of_notes`(*expr*, *big_endian=True*, *iterc-trl=None*)

New in version 1.1. Fill measures in *expr* with minimal number of big-endian notes.

```
>>> measure = Measure((5, 18), [])
```

```
>>> measuretools.fill_measures_in_expr_with_minimal_number_of_notes(
...     measure, big_endian=True)
```

```
>>> f(measure)
{
    ime 5/18
    \scaleDurations #'(8 . 9) {
        c'4 ~
        c'16
    }
}

```

Fill measures in *expr* with minimal number of little-endian notes.

```
>>> measure = Measure((5, 18), [])
```

```
>>> measuretools.fill_measures_in_expr_with_minimal_number_of_notes(
...     measure, big_endian=False)
```

```
>>> f(measure)
{
    ime 5/18
    \scaleDurations #'(8 . 9) {
        c'16 ~
        c'4
    }
}

```

Return none.

measuretools.fill_measures_in_expr_with_repeated_notes

`measuretools.fill_measures_in_expr_with_repeated_notes` (*expr*, *written_duration*, *iter-ctrl=None*)

New in version 1.1. Fill measures in *expr* with repeated notes.

measuretools.fill_measures_in_expr_with_time_signature_denominator_notes

`measuretools.fill_measures_in_expr_with_time_signature_denominator_notes` (*expr*, *iter-ctrl=None*)

New in version 1.1. Fill measures in *expr* with meter denominator notes:

```
>>> staff = Staff([Measure((3, 4), []), Measure((3, 16), []), Measure((3, 8), [])])
>>> measuretools.fill_measures_in_expr_with_time_signature_denominator_notes(staff)
```

```
>>> f(staff)
\new Staff {
  {
    \time 3/4
    c'4
    c'4
    c'4
  }
  {
    \time 3/16
    c'16
    c'16
    c'16
  }
  {
    \time 3/8
    c'8
    c'8
    c'8
  }
}
```

Delete existing contents of measures in *expr*.

Return none.

measuretools.fuse_contiguous_measures_in_container_cyclically_by_counts

`measuretools.fuse_contiguous_measures_in_container_cyclically_by_counts` (*container*, *counts*, *mark=False*)

New in version 1.1. Fuse contiguous measures in *container* cyclically by *counts*:

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 5)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
```

```
>>> f(staff)
\new Staff {
  {
```

```

        \time 2/8
        c'8
        d'8
    }
    {
        e'8
        f'8
    }
    {
        g'8
        a'8
    }
    {
        b'8
        c''8
    }
    {
        d''8
        e''8
    }
}

>>> counts = (2, 1)
>>> measuretools.fuse_contiguous_measures_in_container_cyclically_by_counts(staff, counts)

>>> f(staff)
\new Staff {
    {
        \time 4/8
        c'8
        d'8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
    {
        \time 4/8
        b'8
        c''8
        d''8
        e''8
    }
}

```

Return none.

Set *mark* to true to mark fused measures for later reference. Changed in version 2.0: renamed `fuse.measures_by_counts_cyclic()` to `measuretools.fuse_contiguous_measures_in_container_cyclically_by_counts()`.

measuretools.fuse_measures

`measuretools.fuse_measures(measures)`

New in version 1.1. Fuse *measures*:

```
>>> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (2, 16)]))
>>> measuretools.fill_measures_in_expr_with_repeated_notes(staff, Duration(1, 16))
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> beamtools.BeamSpanner(staff.leaves)
BeamSpanner(c'16, d'16, e'16, f'16)

>>> f(staff)
\new Staff {
    {
        \time 1/8
        c'16 [
        d'16
    ]
    {
        \time 2/16
        e'16
        f'16 ]
    }
}

>>> measuretools.fuse_measures(staff[:])
Measure(2/8, [c'16, d'16, e'16, f'16])

>>> f(staff)
\new Staff {
    {
        \time 2/8
        c'16 [
        d'16
        e'16
        f'16 ]
    }
}
```

Return new measure.

Allow parent-contiguous *measures*.

Allow outside-of-score *measures*.

Do not define measure fusion across intervening container boundaries.

Calculate best new time signature.

Instantiate new measure.

Give *measures* contents to new measure.

Give *measures* dominant spanners to new measure.

Give *measures* parentage to new measure.

Leave *measures* empty, unspanned and outside-of-score. Changed in version 2.0: renamed `fuse.measures_by_reference()` to `measuretools.fuse_measures()`.

measuretools.get_first_measure_in_improper_parentage_of_component

measuretools.get_first_measure_in_improper_parentage_of_component (*component*)

New in version 2.0. Get first measure in improper parentage of *component*:

```
>>> measure = Measure((2, 4), "c'8 d'8 e'8 f'8")
>>> staff = Staff([measure])

>>> f(staff)
\new Staff {
  {
    \time 2/4
    c'8
    d'8
    e'8
    f'8
  }
}

>>> measuretools.get_first_measure_in_improper_parentage_of_component(staff.leaves[0])
Measure(2/4, [c'8, d'8, e'8, f'8])
```

Return measure or none.

measuretools.get_first_measure_in_proper_parentage_of_component

measuretools.get_first_measure_in_proper_parentage_of_component (*component*)

New in version 2.0. Get first measure in proper parentage of *component*:

```
>>> measure = Measure((2, 4), "c'8 d'8 e'8 f'8")
>>> staff = Staff([measure])

>>> f(staff)
\new Staff {
  {
    \time 2/4
    c'8
    d'8
    e'8
    f'8
  }
}

>>> measuretools.get_first_measure_in_proper_parentage_of_component(staff.leaves[0])
Measure(2/4, [c'8, d'8, e'8, f'8])
```

Return measure or none.

measuretools.get_next_measure_from_component

measuretools.get_next_measure_from_component (*component*)

New in version 1.1. Get next measure from *component*.

When *component* is voice, staff or other sequential context, and when *component* contains a measure, return first measure in *component*. This starts the process of forwards measure iteration.

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> measuretools.get_next_measure_from_component(staff)
Measure(2/8, [c'8, d'8])
```

When *component* is voice, staff or other sequential context, and when *component* contains no measure, raise missing measure error.

When *component* is a measure and there is a measure immediately following *component*, return measure immediately following component.

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> measuretools.get_previous_measure_from_component(staff[0]) is None
True
```

When *component* is a measure and there is no measure immediately following *component*, return None.

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> measuretools.get_previous_measure_from_component(staff[-1])
Measure(2/8, [c'8, d'8])
```

When *component* is a leaf and there is a measure in the parentage of *component*, return the measure in the parentage of *component*.

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> measuretools.get_previous_measure_from_component(staff.leaves[0])
Measure(2/8, [c'8, d'8])
```

When *component* is a leaf and there is no measure in the parentage of *component*, raise missing measure error. Changed in version 2.0: renamed `iterate.measure_next()` to `measuretools.get_next_measure_from_component()`.

measuretools.get_nth_measure_in_expr

`measuretools.get_nth_measure_in_expr(expr, n=0)`

New in version 2.0. Get nth measure in *expr*:

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
}
```

```
}
}
```

Read forward for positive values of n .

```
>>> for n in range(3):
...     measuretools.get_nth_measure_in_expr(staff, n)
...
Measure(2/8, [c'8, d'8])
Measure(2/8, [e'8, f'8])
Measure(2/8, [g'8, a'8])
```

Read backward for negative values of n .

```
>>> for n in range(3, -1, -1):
...     measuretools.get_nth_measure_in_expr(staff, n)
...
Measure(2/8, [g'8, a'8])
Measure(2/8, [e'8, f'8])
Measure(2/8, [c'8, d'8])
```

Changed in version 2.0: renamed `iterate.get_nth_measure()` to `measuretools.get_nth_measure_in_expr()`.

`measuretools.get_one_indexed_measure_number_in_expr`

`measuretools.get_one_indexed_measure_number_in_expr(expr, measure_number)`

New in version 2.0. Get one-indexed *measure_number* in *expr*:

```
>>> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(t)

>>> f(t)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
}
>>> measuretools.get_one_indexed_measure_number_in_expr(t, 3)
Measure(2/8, [g'8, a'8])
```

Note that measures number from 1.

`measuretools.get_previous_measure_from_component`

`measuretools.get_previous_measure_from_component(component)`

New in version 1.1. Get previous measure from *component*.

When *component* is voice, staff or other sequential context, and when *component* contains a measure, return last measure in *component*. This starts the process of backwards measure iteration.

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> measuretools.get_previous_measure_from_component(staff)
Measure(2/8, [e'8, f'8])
```

When *component* is voice, staff or other sequential context, and when *component* contains no measure, raise missing measure error.

When *component* is a measure and there is a measure immediately preceeding *component*, return measure immediately preceeding component.

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> measuretools.get_previous_measure_from_component(staff[-1])
Measure(2/8, [c'8, d'8])
```

When *component* is a measure and there is no measure immediately preceeding *component*, return None.

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> measuretools.get_previous_measure_from_component(staff[0]) is None
True
```

When *component* is a leaf and there is a measure in the parentage of *component*, return the measure in the parentage of *component*.

```
>>> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)
>>> measuretools.get_previous_measure_from_component(staff.leaves[0])
Measure(2/8, [c'8, d'8])
```

When *component* is a leaf and there is no measure in the parentage of *component*, raise missing measure error. Changed in version 2.0: renamed `iterate.measure_prev()` to `measuretools.get_previous_measure_from_component()`.

measuretools.list_time_signatures_of_measures_in_expr

`measuretools.list_time_signatures_of_measures_in_expr(components)`

New in version 2.0. List time signatures of measures in *expr*:

```
>>> staff = Staff('abj: | 2/8 c8 d8 || 3/8 c8 d8 e8 || 4/8 c8 d8 e8 f8 |')

>>> f(staff)
\new Staff {
  {
    \time 2/8
    c8
    d8
  }
  {
    \time 3/8
    c8
    d8
    e8
  }
}
```

```

    {
        \time 4/8
        c8
        d8
        e8
        f8
    }
}

>>> for x in measuretools.list_time_signatures_of_measures_in_expr(staff): x
...
TimeSignatureMark((2, 8))(|2/8, c8, d8|)
TimeSignatureMark((3, 8))(|3/8, c8, d8, e8|)
TimeSignatureMark((4, 8))(|4/8, c8, d8, e8, f8|)

```

Return list of zero or more time signatures. Changed in version 2.0: re-named `measuretools.list_time_signatures_of_mesures_in_expr()` to `measuretools.list_time_signatures_of_measures_in_expr()`.

measuretools.make_measures_with_full_measure_spacer_skips

`measuretools.make_measures_with_full_measure_spacer_skips` (*time_signatures*)
 New in version 1.1. Make measures with full-measure spacer skips from *time_signatures*:

```

>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (5, 16), (5, 16)])

>>> staff = Staff(measures)

>>> f(staff)
\new Staff {
    {
        \time 1/8
        s1 * 1/8
    }
    {
        \time 5/16
        s1 * 5/16
    }
    {
        s1 * 5/16
    }
}

```

Return list of rigid measures. Changed in version 2.0: renamed `measuretools.make()` to `measuretools.make_measures_with_full_measure_spacer_skips()`.

measuretools.measure_to_one_line_input_string

`measuretools.measure_to_one_line_input_string` (*measure*)
 New in version 2.6. Change *measure* to one-line input string:

```

>>> measure = Measure((4, 4), "c'4 d'4 e'4 f'4")

```

```

>>> input_string = measuretools.measure_to_one_line_input_string(measure)

>>> print input_string
Measure((4, 4), "c'4 d'4 e'4 f'4")

>>> new_measure = eval(input_string)

>>> new_measure
Measure(4/4, [c'4, d'4, e'4, f'4])

>>> f(new_measure)
{
    \time 4/4
    c'4
    d'4
    e'4
    f'4
}

```

The purpose of this function is to create an evaluable string from simple measures.

Spanners, articulations and overrides not supported.

Return string.

measuretools.move_full_measure_tuplet_prolation_to_measure_time_signature

measuretools.move_full_measure_tuplet_prolation_to_measure_time_signature (*expr*)

New in version 1.1. Move prolotion of full-measure tuplet to meter of measure.

Measures usually become nonbinary as as result:

```

>>> t = Measure((2, 8), [tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")])
>>> measuretools.move_full_measure_tuplet_prolation_to_measure_time_signature(t)

>>> f(t)
{
    \time 3/12
    \scaleDurations #'(2 . 3) {
        c'8
        d'8
        e'8
    }
}

```

Return none. Changed in version 2.0: renamed `measuretools.subsume()` to `measuretools.move_full_measure_tuplet_prolation_to_measure_time_signature()`.

measuretools.move_measure_prolation_to_full_measure_tuplet

measuretools.move_measure_prolation_to_full_measure_tuplet (*expr*)

New in version 2.0. Move measure prolotion to full-measure tuplet.

Turn nonbinary measures into binary measures containing a single fixed-duration tuplet.

Note that not all nonbinary measures can be made binary.

Returns None because processes potentially many measures. Changed in version 2.0: renamed `measuretools.project()` to `measuretools.move_measure_prolation_to_full_measure_tuplet()`.

`measuretools.multiply_and_scale_contents_of_measures_in_expr`

`measuretools.multiply_and_scale_contents_of_measures_in_expr`(*expr*, *multiplier_pairs*)

New in version 1.1. Multiply and scale contents of measures in *expr* by *multiplier_pairs*.

The *multiplier_pairs* argument must be a list of (*contents_multiplier*, *denominator_multiplier*) pairs.

Both *contents_multiplier* and *denominator_multiplier* must be positive integers.

Example 1. Multiply measure contents by 3. Scale time signature denominator by 3:

```
>>> measure = Measure((3, 16), "c'16 c'16 c'16")

>>> measuretools.multiply_and_scale_contents_of_measures_in_expr(measure, [(3, 3)])
[Measure(9/48, [c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32])]
```

Example 2. Multiply measure contents by 3. Scale time signature denominator by 2:

```
>>> measure = Measure((3, 16), "c'16 c'16 c'16")

>>> measuretools.multiply_and_scale_contents_of_measures_in_expr(measure, [(3, 2)])
[Measure(9/32, [c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32])]
```

Example 3. Multiply measure contents 3:

```
>>> measure = Measure((3, 16), "c'16 c'16 c'16")

>>> measuretools.multiply_and_scale_contents_of_measures_in_expr(measure, [(3, 1)])
[Measure(9/16, [c'16, c'16, c'16, c'16, c'16, c'16, c'16, c'16, c'16])]
```

Return list of measures changed.

`measuretools.multiply_contents_of_measures_in_expr`

`measuretools.multiply_contents_of_measures_in_expr`(*expr*, *n*)

New in version 1.1. Multiply contents *n* - 1 times and adjust meter of every measure in *expr*:

```
>>> measure = Measure((3, 8), "c'8 d'8 e'8")
>>> beamtools.BeamSpanner(measure.leaves)
BeamSpanner(c'8, d'8, e'8)

>>> f(measure)
{
  \time 3/8
  c'8 [
  d'8
  e'8 ]
}

>>> measuretools.multiply_contents_of_measures_in_expr(measure, 3)
```



```
>>> f(measure)
{
    \time 9/8
    c'8 [
    d'8
    e'8 ]
    c'8 [
    d'8
    e'8 ]
    c'8 [
    d'8
    e'8 ]
}
```

Changed in version 2.0: renamed `measuretools.spin()` to `measuretools.multiply_contents_of_measures_`

`measuretools.pad_measures_in_expr_with_rests`

`measuretools.pad_measures_in_expr_with_rests(expr, front, back, splice=False)`

New in version 1.1. Pad measures in *expr* with rests.

Iterate all measures in *expr*. Insert rest with duration equal to *front* at beginning of each measure. Insert rest with duration equal to *back* at end of each measure.

Set *front* to a positive rational or none. Set *back* to a positive rational or none.

Note that this function is designed to help create regularly spaced charts and tables of musical materials. This function makes most sense when used on anonymous measures or dynamic measures.

```
>>> t = Staff(measuretools.AnonymousMeasure("c'8 d'8") * 2)
>>> front, back = Duration(1, 32), Duration(1, 64)
>>> measuretools.pad_measures_in_expr_with_rests(t, front, back)
```

```
>>> f(t)
\new Staff {
    {
        \override Staff.TimeSignature #'stencil = ##f
        \time 19/64
        r32
        c'8
        d'8
        r64
        \revert Staff.TimeSignature #'stencil
    }
    {
        \override Staff.TimeSignature #'stencil = ##f
        r32
        c'8
        d'8
        r64
        \revert Staff.TimeSignature #'stencil
    }
}
```

Works when measures contain stacked voices:

```
>>> measure = measuretools.DynamicMeasure(
...     Voice(notetools.make_repeated_notes(2)) * 2)
```

```
>>> measure.is_parallel = True
>>> t = Staff(measure * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(t)
>>> measuretools.pad_measures_in_expr_with_rests(
...     t, Duration(1, 32), Duration(1, 64))

>>> f(t)
\new Staff {
  <<
    \time 19/64
    \new Voice {
      r32
      c'8
      d'8
      r64
    }
    \new Voice {
      r32
      e'8
      f'8
      r64
    }
  >>
  <<
    \new Voice {
      r32
      g'8
      a'8
      r64
    }
    \new Voice {
      r32
      b'8
      c''8
      r64
    }
  >>
}
```

Set the optional *splice* keyword to `True` to extend edge spanners over newly inserted rests:

```
>>> t = measuretools.DynamicMeasure("c'8 d'8")
>>> beamtools.BeamSpanner(t[:])
BeamSpanner(c'8, d'8)
>>> measuretools.pad_measures_in_expr_with_rests(
...     t, Duration(1, 32), Duration(1, 64), splice = True)

>>> f(t)
{
  \time 19/64
  r32 [
  c'8
  d'8
  r64 ]
}
```

Return `none`.

Raise value when *front* is neither a positive rational nor `none`.

Raise value when *back* is neither a positive rational nor none. Changed in version 2.0: renamed `layout.insert_measure_padding_rest()` to `measuretools.pad_measures_in_expr_with_rests()`.

`measuretools.pad_measures_in_expr_with_skips`

`measuretools.pad_measures_in_expr_with_skips(expr, front, back, splice=False)`

New in version 2.0. Pad measures in *expr* with skips.

Iterate all measures in *expr*. Insert skip with duration equal to *front* at beginning of each measure. Insert skip with duration equal to *back* at end of each measure.

Set *front* to a positive rational or none. Set *back* to a positive rational or none.

Note that this function is designed to help create regularly spaced charts and tables of musical materials. This function makes most sense when used on anonymous measures and dynamic measures.

```
>>> t = Staff(measuretools.AnonymousMeasure("c'8 d'8") * 2)
>>> front, back = Duration(1, 32), Duration(1, 64)
>>> measuretools.pad_measures_in_expr_with_skips(t, front, back)
```

```
>>> f(t)
\new Staff {
  {
    \override Staff.TimeSignature #'stencil = ##f
    \time 19/64
    s32
    c'8
    d'8
    s64
    \revert Staff.TimeSignature #'stencil
  }
  {
    \override Staff.TimeSignature #'stencil = ##f
    s32
    c'8
    d'8
    s64
    \revert Staff.TimeSignature #'stencil
  }
}
```

Works when measures contain stacked voices.

```
>>> measure = measuretools.DynamicMeasure(
...     Voice(notetools.make_repeated_notes(2)) * 2)
>>> measure.is_parallel = True
>>> t = Staff(measure * 2)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(t)
>>> measuretools.pad_measures_in_expr_with_skips(
...     t, Duration(1, 32), Duration(1, 64))

>>> f(t)
\new Staff {
  <<
    \time 19/64
    \new Voice {
      s32
```

```

        c'8
        d'8
        s64
    }
    \new Voice {
        s32
        e'8
        f'8
        s64
    }
>>
<<
    \new Voice {
        s32
        g'8
        a'8
        s64
    }
    \new Voice {
        s32
        b'8
        c''8
        s64
    }
>>
}

```

Set the optional *splice* keyword to `True` to extend edge spanners over newly inserted skips:

```

>>> t = measuretools.DynamicMeasure("c'8 d'8")
>>> beamtools.BeamSpanner(t[:])
BeamSpanner(c'8, d'8)
>>> measuretools.pad_measures_in_expr_with_skips(
...     t, Duration(1, 32), Duration(1, 64), splice = True)

>>> f(t)
{
    \time 19/64
    s32 [
    c'8
    d'8
    s64 ]
}

```

Return `none`.

Raise value error when *front* is neither a positive rational nor `none`.

Raise value error when *back* is neither a positive rational nor `none`. Changed in version 2.0: renamed `layout.insert_measure_padding_skip()` to `measuretools.pad_measures_in_expr_with_skips()`.

measuretools.replace_contents_of_measures_in_expr

`measuretools.replace_contents_of_measures_in_expr(expr, new_contents)`

New in version 1.1. Replace contents of measures in *expr* with *new_contents*:

```
>>> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (3, 16)]))

>>> f(staff)
\new Staff {
    {
        \time 1/8
        s1 * 1/8
    }
    {
        \time 3/16
        s1 * 3/16
    }
}

>>> notes = [Note("c'16"), Note("d'16"), Note("e'16"), Note("f'16")]
>>> measuretools.replace_contents_of_measures_in_expr(staff, notes)
[Measure(1/8, [c'16, d'16]), Measure(3/16, [e'16, f'16, s1 * 1/16])]

>>> f(staff)
\new Staff {
    {
        \time 1/8
        c'16
        d'16
    }
    {
        \time 3/16
        e'16
        f'16
        s1 * 1/16
    }
}
```

Preserve duration of all measures.

Skip measures that are too small.

Pad extra space at end of measures with spacer skip.

If not enough measures raise stop iteration.

Return measures iterated. Changed in version 2.0: renamed `measuretools.overwrite_contents()` to `measuretools.replace_contents_of_measures_in_expr()`.

measuretools.report_time_signature_distribution

`measuretools.report_time_signature_distribution(expr)`

New in version 2.0. Report meter distribution of *expr* as string:

```
>>> measuretools.report_time_signature_distribution(t)
'\t3/80\t2\n\t2/16\t73\n\t7/40\t1\n\t3/16\t20\n\t16/80\t1\n\t17/80\t1\n\n\t19/80\t1\n\t4/16\t73\n\t5/16\t62\n\t13/40\t1\n\t27/80\t1\n\t6/16\t12\n\n\t7/16\t16\n\t8/16\t13\n\t9/16\t15\n\t10/16\t4\n'
```

Return string.

measuretools.scale_contents_of_measures_in_expr

`measuretools.scale_contents_of_measures_in_expr(expr, multiplier=1)`

New in version 2.0. Scale contents of measures in *expr* by *multiplier*.

Iterate *expr*. For every measure in *expr* first multiply the measure meter by *multiplier* and then scale measure contents to fit the new meter.

Extend `containertools.scale_contents_of_container()`.

Return *none*.

measuretools.scale_measure_and_adjust_time_signature

`measuretools.scale_measure_and_adjust_time_signature(measure, multiplier=1)`

New in version 2.0. Scale *measure* by *multiplier* and adjust meter:

```
>>> t = Measure((3, 8), "c'8 d'8 e'8")
>>> measuretools.scale_measure_and_adjust_time_signature(t, Duration(2, 3))
Measure(3/12, [c'8, d'8, e'8])
```

```
>>> f(t)
{
  \time 3/12
  \scaleDurations #'(2 . 3) {
    c'8
    d'8
    e'8
  }
}
```

Return *measure*.

measuretools.scale_measure_denominator_and_adjust_measure_contents

`measuretools.scale_measure_denominator_and_adjust_measure_contents(measure, new_denominator_factor)`

New in version 1.1. Change binary *measure* to nonbinary measure with *new_denominator_factor*:

```
>>> measure = Measure((2, 8), "c'8 d'8")
>>> beamtools.BeamSpanner(measure.leaves)
BeamSpanner(c'8, d'8)

>>> f(measure)
{
  \time 2/8
  c'8 [
  d'8 ]
}

>>> measuretools.scale_measure_denominator_and_adjust_measure_contents(measure, 3)
Measure(3/12, [c'8., d'8.])

>>> f(measure)
{
  \time 3/12
  \scaleDurations #'(2 . 3) {
```

```

        c'8. [
        d'8. ]
    }
}

```

Treat *new_denominator_factor* like clever form of 1: 3/3 or 5/5 or 7/7, etc.

Preserve *measure* prolated duration.

Derive new *measure* multiplier.

Scale *measure* contents.

Pick best new meter. Changed in version 2.0: renamed `measuretools.change_binary_measure_to_nonbinary()` to `measuretools.scale_measure_denominator_and_adjust_measure_contents()`.

`measuretools.set_always_format_time_signature_of_measures_in_expr`

`measuretools.set_always_format_time_signature_of_measures_in_expr(expr, value=True)`

New in version 2.9. Set *always_format_time_signature* of measures in *expr* to boolean *value*.

Return none.

`measuretools.set_measure_denominator_and_adjust_numerator`

`measuretools.set_measure_denominator_and_adjust_numerator(measure, denominator)`

New in version 1.1. Set *measure* meter *denominator* and multiply meter numerator accordingly:

```

>>> measure = Measure((3, 8), "c'8 d'8 e'8")
>>> beamtools.BeamSpanner(measure.leaves)
BeamSpanner(c'8, d'8, e'8)

>>> f(measure)
{
    \time 3/8
    c'8 [
    d'8
    e'8 ]
}

>>> measuretools.set_measure_denominator_and_adjust_numerator(measure, 16)
Measure(6/16, [c'8, d'8, e'8])

>>> f(measure)
{
    \time 6/16
    c'8 [
    d'8
    e'8 ]
}

```

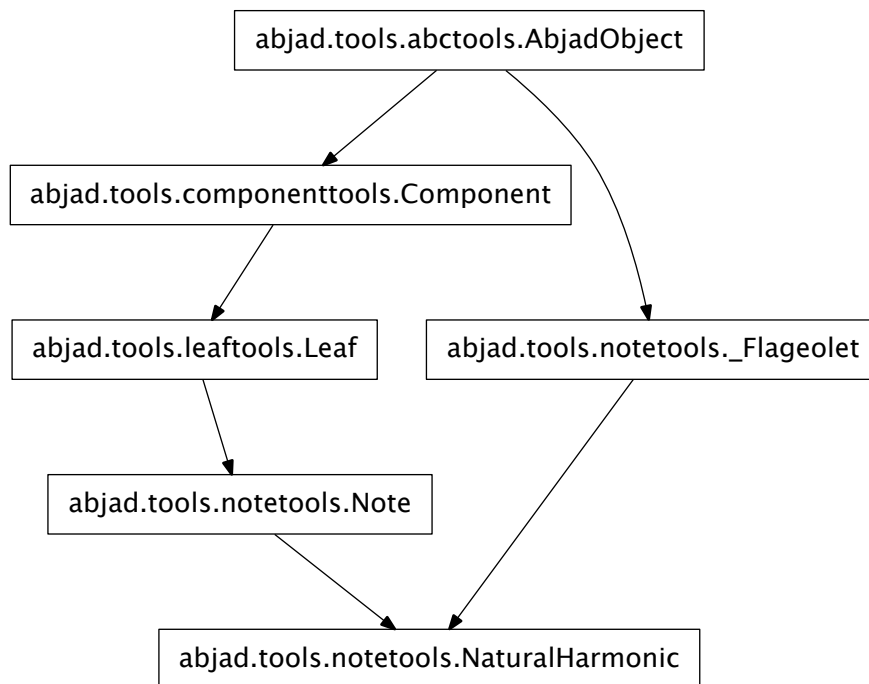
Leave *measure* contents unchanged.

Return *measure*. Changed in version 2.0: renamed `measuretools.set_measure_denominator_and_multiply_numerator()` to `measuretools.set_measure_denominator_and_adjust_numerator()`.

notetools

concrete classes

notetools.NaturalHarmonic



```

class notetools.NaturalHarmonic(*args)
    Abjad model of natural harmonic.

    Initialize natural harmonic by hand:

    >>> notetools.NaturalHarmonic("cs'8.")
    NaturalHarmonic(cs', 8.)

    Initialize natural harmonic from note:

    >>> note = Note("cs'8.")

    >>> notetools.NaturalHarmonic(note)
    NaturalHarmonic(cs', 8.)

    Natural harmonics are immutable.
  
```

Read-only Properties

`NaturalHarmonic.duration_in_seconds`
 Inherited from `leaftools.Leaf`

NaturalHarmonic.fingered_pitch

Read-only fingered pitch of note:

```
>>> staff = Staff("d''8 e''8 f''8 g''8")
>>> piccolo = instrumenttools.Piccolo()(staff)
>>> instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch(staff)

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Piccolo }
  \set Staff.shortInstrumentName = \markup { Picc. }
  d'8
  e'8
  f'8
  g'8
}

>>> staff[0].fingered_pitch
NamedChromaticPitch("d' ")
```

Return named chromatic pitch.

Inherited from `notetools.Note`**NaturalHarmonic.leaf_index**Inherited from `leaftools.Leaf`**NaturalHarmonic.lilypond_format**Inherited from `componenttools.Component`**NaturalHarmonic.multiplied_duration**Inherited from `leaftools.Leaf`**NaturalHarmonic.override**

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`**NaturalHarmonic.parent**Inherited from `componenttools.Component`**NaturalHarmonic.preprolated_duration**Inherited from `leaftools.Leaf`**NaturalHarmonic.prolated_duration**Inherited from `componenttools.Component`**NaturalHarmonic.prolation**Inherited from `componenttools.Component`**NaturalHarmonic.set**

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`**NaturalHarmonic.sounding_pitch**

Read-only sounding pitch of note:

```
>>> staff = Staff("d''8 e''8 f''8 g''8")
>>> piccolo = instrumenttools.Piccolo()(staff)

>>> instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch(staff)
```

```
>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Piccolo }
  \set Staff.shortInstrumentName = \markup { Picc. }
  d'8
  e'8
  f'8
  g'8
}
>>> staff[0].sounding_pitch
NamedChromaticPitch("d'")
```

Return named chromatic pitch.

Inherited from `notetools.Note`

`NaturalHarmonic.spanners`

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`NaturalHarmonic.start_offset`

Read-only start offset of component.

Inherited from `componenttools.Component`

`NaturalHarmonic.start_offset_in_seconds`

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

`NaturalHarmonic.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`NaturalHarmonic.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`NaturalHarmonic.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`NaturalHarmonic.suono_reale`

Actual sound of the harmonic when played.

Inherited from `notetools._Flageolet`

Read/write Properties

`NaturalHarmonic.duration_multiplier`

Inherited from `leaftools.Leaf`

`NaturalHarmonic.note_head`

Get note head of note:

```
>>> note = Note(13, (3, 16))
>>> note.note_head
NoteHead("cs'")
```

Set note head of note:

```
>>> note = Note(13, (3, 16))
>>> note.note_head = 14
>>> note
Note("d' '8.")
```

Inherited from `notetools.Note`

`NaturalHarmonic.written_duration`

Inherited from `leaftools.Leaf`

`NaturalHarmonic.written_pitch`

Get named pitch of note:

```
>>> note = Note(13, (3, 16))
>>> note.written_pitch
NamedChromaticPitch("cs' '")
```

Set named pitch of note:

```
>>> note = Note(13, (3, 16))
>>> note.written_pitch = 14
>>> note
Note("d' '8.")
```

Inherited from `notetools.Note`

`NaturalHarmonic.written_pitch_indication_is_at_sounding_pitch`

Inherited from `leaftools.Leaf`

`NaturalHarmonic.written_pitch_indication_is_nonsemantic`

Inherited from `leaftools.Leaf`

Special Methods

`NaturalHarmonic.__and__(arg)`

Inherited from `leaftools.Leaf`

`NaturalHarmonic.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__mul__(n)`

Inherited from `componenttools.Component`

`NaturalHarmonic.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__or__(arg)`

Inherited from `leaftools.Leaf`

`NaturalHarmonic.__repr__()`

`NaturalHarmonic.__rmul__(n)`

Inherited from `componenttools.Component`

`NaturalHarmonic.__str__()`

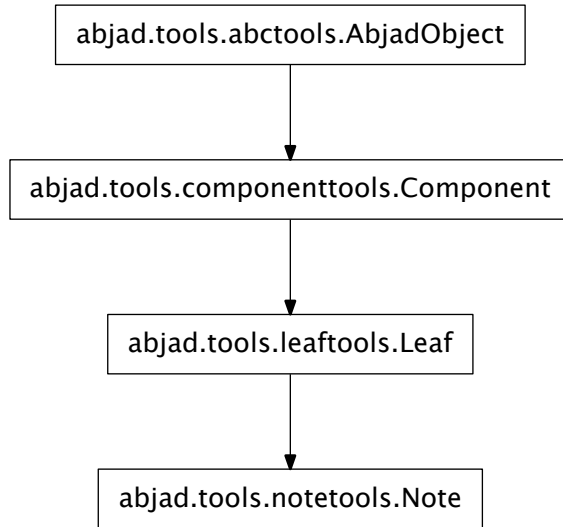
Inherited from `leaftools.Leaf`

`NaturalHarmonic.__sub__(arg)`

Inherited from `leaftools.Leaf`

`NaturalHarmonic.__xor__(arg)`

Inherited from `leaftools.Leaf`

notetools.Note

class notetools.**Note** (*args, **kwargs)
 New in version 1.0. Abjad model of a note:

```
>>> Note("cs'8.")
Note("cs'8.")
```

Notes are immutable.

Read-only Properties

Note.duration_in_seconds

Inherited from leaftools.Leaf

Note.fingered_pitch

Read-only fingered pitch of note:

```
>>> staff = Staff("d'8 e'8 f'8 g'8")
>>> piccolo = instrumenttools.Piccolo()(staff)
>>> instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch(staff)

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Piccolo }
  \set Staff.shortInstrumentName = \markup { Picc. }
  d'8
  e'8
  f'8
  g'8
}
```

```
>>> staff[0].fingered_pitch
NamedChromaticPitch("d' ")
```

Return named chromatic pitch.

Note.**leaf_index**

Inherited from `leaftools.Leaf`

Note.**lilypond_format**

Inherited from `componenttools.Component`

Note.**multiplied_duration**

Inherited from `leaftools.Leaf`

Note.**override**

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Note.**parent**

Inherited from `componenttools.Component`

Note.**preprolated_duration**

Inherited from `leaftools.Leaf`

Note.**prolated_duration**

Inherited from `componenttools.Component`

Note.**prolation**

Inherited from `componenttools.Component`

Note.**set**

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Note.**sounding_pitch**

Read-only sounding pitch of note:

```
>>> staff = Staff("d''8 e''8 f''8 g''8")
>>> piccolo = instrumenttools.Piccolo()(staff)

>>> instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch(staff)

>>> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Piccolo }
  \set Staff.shortInstrumentName = \markup { Picc. }
  d'8
  e'8
  f'8
  g'8
}
>>> staff[0].sounding_pitch
NamedChromaticPitch("d' ")
```

Return named chromatic pitch.

Note.**spanners**

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Note.**start_offset**

Read-only start offset of component.

Inherited from `componenttools.Component`

Note.**start_offset_in_seconds**

Read-only start offset of comonent in seconds.

Inherited from `componenttools.Component`

Note.**stop_offset**

Read-only stop offset of component.

Inherited from `componenttools.Component`

Note.**stop_offset_in_seconds**

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Note.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Note.**duration_multiplier**

Inherited from `leaftools.Leaf`

Note.**note_head**

Get note head of note:

```
>>> note = Note(13, (3, 16))
>>> note.note_head
NoteHead("cs' ' ")
```

Set note head of note:

```
>>> note = Note(13, (3, 16))
>>> note.note_head = 14
>>> note
Note("d' ' 8.")
```

Note.**written_duration**

Inherited from `leaftools.Leaf`

Note.**written_pitch**

Get named pitch of note:

```
>>> note = Note(13, (3, 16))
>>> note.written_pitch
NamedChromaticPitch("cs' ' ")
```

Set named pitch of note:

```
>>> note = Note(13, (3, 16))
>>> note.written_pitch = 14
>>> note
Note("d' ' 8.")
```

Note.**written_pitch_indication_is_at_sounding_pitch**

Inherited from `leaftools.Leaf`

Note.**written_pitch_indication_is_nonsemantic**

Inherited from `leaftools.Leaf`

Special Methods

Note.**__and__**(*arg*)

Inherited from `leaftools.Leaf`

Note.**__eq__**(*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Note.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Note.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Note.**__le__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Note.**__lt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Note.**__mul__**(*n*)

Inherited from `componenttools.Component`

Note.**__ne__**(*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Note.**__or__**(*arg*)

Inherited from `leaftools.Leaf`

Note.**__repr__**()

Inherited from `leaftools.Leaf`

Note.**__rmul__**(*n*)

Inherited from `componenttools.Component`

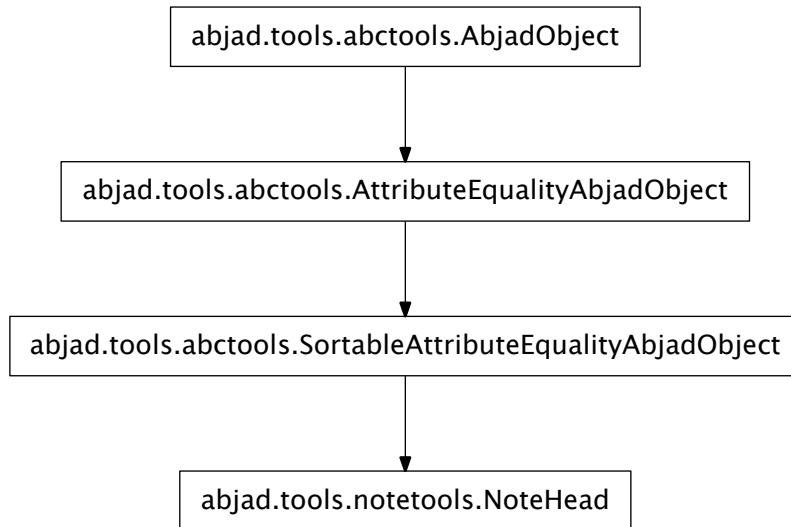
Note.**__str__**()

Inherited from `leaftools.Leaf`

`Note.__sub__(arg)`
Inherited from `leaftools.Leaf`

`Note.__xor__(arg)`
Inherited from `leaftools.Leaf`

`notetools.NoteHead`



class `notetools.NoteHead`(*written_pitch=None, client=None, is_cautionary=False, is_forced=False, tweak_pairs=()*)

Abjad model of a note head:

```
>>> notetools.NoteHead(13)
NoteHead("cs' ' ")
```

Note heads are immutable.

Read-only Properties

`NoteHead.lilypond_format`

Read-only LilyPond input format of note head:

```
>>> note_head = notetools.NoteHead("cs' ' ")
>>> note_head.lilypond_format
"cs' ' "
```

Return string.

`NoteHead.named_chromatic_pitch`

Read-only named chromatic pitch equal to note head:

```
>>> note_head = notetools.NoteHead("cs'")
>>> note_head.named_chromatic_pitch
NamedChromaticPitch("cs'")
```

Return named chromatic pitch.

NoteHead.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

NoteHead.tweak

Read-only LilyPond tweak reservoir:

```
>>> note_head = notetools.NoteHead("cs'")
>>> note_head.tweak
LilyPondTweakReservoir()
```

Return LilyPond tweak reservoir.

Read/write Properties

NoteHead.is_cautionary

Get cautionary accidental flag:

```
>>> note_head = notetools.NoteHead("cs'")
>>> note_head.is_cautionary
False
```

Set cautionary accidental flag:

```
>>> note_head = notetools.NoteHead("cs'")
>>> note_head.is_cautionary = True
```

Return boolean.

NoteHead.is_forced

NoteHead.written_pitch

Get named pitch of note head:

```
>>> note_head = notetools.NoteHead("cs'")
>>> note_head.written_pitch
NamedChromaticPitch("cs'")
```

Set named pitch of note head:

```
>>> note_head = notetools.NoteHead("cs'")
>>> note_head.written_pitch = "d'"
>>> note_head.written_pitch
NamedChromaticPitch("d'")
```

Set pitch token.

Special Methods

NoteHead.__eq__(*arg*)

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

```
NoteHead.__ge__(arg)
    Initialize new object from arg and evaluate comparison attributes.

    Return boolean.

    Inherited from abctools.SortableAttributeEqualityAbjadObject

NoteHead.__gt__(arg)
    Initialize new object from arg and evaluate comparison attributes.

    Return boolean.

    Inherited from abctools.SortableAttributeEqualityAbjadObject

NoteHead.__le__(arg)
    Initialize new object from arg and evaluate comparison attributes.

    Return boolean.

    Inherited from abctools.SortableAttributeEqualityAbjadObject

NoteHead.__lt__(arg)
    Initialize new object from arg and evaluate comparison attributes.

    Return boolean.

    Inherited from abctools.SortableAttributeEqualityAbjadObject

NoteHead.__ne__(arg)
    Initialize new object from arg and evaluate comparison attributes.

    Return boolean.

    Inherited from abctools.AttributeEqualityAbjadObject

NoteHead.__repr__()

NoteHead.__str__()
```

functions

notetools.add_artificial_harmonic_to_note

```
notetools.add_artificial_harmonic_to_note(note, melodic_diatonic_interval=None)
    Add artifical harmonic to note at melodic_diatonic_interval:

    >>> staff = Staff("c'8 d'8 e'8 f'8")
    >>> beamtools.BeamSpanner(staff[:])
    BeamSpanner(c'8, d'8, e'8, f'8)

    >>> f(staff)
    \new Staff {
      c'8 [
      d'8
      e'8
      f'8 ]
    }

    >>> notetools.add_artificial_harmonic_to_note(staff[0])
    Chord("<c' f'>8")
```

```
>>> f(staff)
\new Staff {
  <
    c'
    \tweak #'style #'harmonic
    f'
  >8 [
    d'8
    e'8
    f'8 ]
}
```

When `melodic_diatonic_interval=None` set to a perfect fourth.

Create new artificial harmonic chord from *note*.

Move parentage and spanners from *note* to artificial harmonic chord.

Return artificial harmonic chord. Changed in version 2.0: renamed `harmonictools.add_artificial()` to `notetools.add_artificial_harmonic_to_note()`.

notetools.all_are_notes

`notetools.all_are_notes(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad notes:

```
>>> notes = [Note("c'4"), Note("d'4"), Note("e'4")]
```

```
>>> notetools.all_are_notes(notes)
True
```

True when *expr* is an empty sequence:

```
>>> notetools.all_are_notes([])
True
```

Otherwise false:

```
>>> notetools.all_are_notes('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

notetools.make_accelerating_notes_with_lilypond_multipliers

`notetools.make_accelerating_notes_with_lilypond_multipliers(pitches, total, start, stop, exp='cosine', written='8')`

Make accelerating notes with LilyPond multipliers:

```
>>> pitches = [1, 2]
>>> total = (1, 2)
>>> start = (1, 4)
>>> stop = (1, 8)
>>> args = [pitches, total, start, stop]
```

```
>>> notes = notetools.make_accelerating_notes_with_lilypond_multipliers(*args)

>>> notes
[Note("cs'8 * 113/64"), Note("d'8 * 169/128"), Note("cs'8 * 117/128")]

>>> Voice(notes).prolated_duration
Duration(1, 2)
```

Set note pitches cyclically from *pitches*.

Return as many interpolation values as necessary to fill the *total* duration requested.

Interpolate durations from *start* to *stop*.

Set note durations to *written* duration times computed interpolated multipliers.

Return list of notes. Changed in version 2.0: renamed `construct.notes_curve()` to `notetools.make_accelerating_notes_with_lilypond_multipliers()`.

notetools.make_notes

`notetools.make_notes(pitches, durations, big_endian=True)`

Make notes according to *pitches* and *durations*.

Cycle through *pitches* when the length of *pitches* is less than the length of *durations*:

```
>>> notetools.make_notes([0], [(1, 16), (1, 8), (1, 8)])
[Note("c'16"), Note("c'8"), Note("c'8")]
```

Cycle through *durations* when the length of *durations* is less than the length of *pitches*:

```
>>> notetools.make_notes([0, 2, 4, 5, 7], [(1, 16), (1, 8), (1, 8)])
[Note("c'16"), Note("d'8"), Note("e'8"), Note("f'16"), Note("g'8")]
```

Create ad hoc tuplets for nonassignable durations:

```
>>> notetools.make_notes([0], [(1, 16), (1, 12), (1, 8)])
[Note("c'16"), Tuplet(2/3, [c'8]), Note("c'8")]
```

Set `big_endian=True` to express tied values in decreasing duration:

```
>>> notetools.make_notes([0], [(13, 16)], big_endian=True)
[Note("c'2."), Note("c'16")]
```

Set `big_endian=False` to express tied values in increasing duration:

```
>>> notetools.make_notes([0], [(13, 16)], big_endian=False)
[Note("c'16"), Note("c'2.")]
```

Set *pitches* to a single pitch or a sequence of pitches.

Set *durations* to a single duration or a list of durations.

Return list of newly constructed notes. Changed in version 2.0: renamed `construct.notes()` to `notetools.make_notes()`.

notetools.make_notes_with_multiplied_durations

`notetools.make_notes_with_multiplied_durations` (*pitch*, *written_duration*, *multiplied_durations*)

New in version 2.0. Make *written_duration* notes with *pitch* and *multiplied_durations*:

```
>>> args = [0, Duration(1, 4), [(1, 2), (1, 3), (1, 4), (1, 5)]]
>>> notetools.make_notes_with_multiplied_durations(*args)
[Note("c'4 * 2"), Note("c'4 * 4/3"), Note("c'4 * 1"), Note("c'4 * 4/5")]
```

Useful for making spatially positioned notes.

Return list of notes.

notetools.make_percussion_note

`notetools.make_percussion_note` (*pitch*, *total_duration*, *max_note_duration*=(1, 8))

Make percussion note:

```
>>> notetools.make_percussion_note(2, (1, 4), (1, 8))
[Note("d'8"), Rest('r8')]

>>> notetools.make_percussion_note(2, (1, 64), (1, 8))
[Note("d'64")]

>>> notetools.make_percussion_note(2, (5, 64), (1, 8))
[Note("d'16"), Rest('r64')]

>>> notetools.make_percussion_note(2, (5, 4), (1, 8))
[Note("d'8"), Rest('r1'), Rest('r8')]
```

Return list of newly constructed note followed by zero or more newly constructed rests.

Durations of note and rests returned will sum to *total_duration*.

Duration of note returned will be no greater than *max_note_duration*.

Duration of rests returned will sum to note duration taken from *total_duration*.

Useful for percussion music where attack duration is negligible and tied notes undesirable. Changed in version 2.0: renamed `construct.percussion_note()` to `notetools.make_percussion_note()`.

notetools.make_quarter_notes_with_lilypond_multipliers

`notetools.make_quarter_notes_with_lilypond_multipliers` (*pitches*, *multiplied_durations*)

New in version 2.0. Make quarter notes with *pitches* and *multiplied_durations*:

```
>>> args = [[0, 2, 4, 5], [(1, 4), (1, 5), (1, 6), (1, 7)]]
>>> notetools.make_quarter_notes_with_lilypond_multipliers(*args)
[Note("c'4 * 1"), Note("d'4 * 4/5"), Note("e'4 * 2/3"), Note("f'4 * 4/7")]
```

Read *pitches* cyclically where the length of *pitches* is less than the length of *multiplied_durations*:

```
>>> args = [[0], [(1, 4), (1, 5), (1, 6), (1, 7)]]
>>> notetools.make_quarter_notes_with_lilypond_multipliers(*args)
[Note("c'4 * 1"), Note("c'4 * 4/5"), Note("c'4 * 2/3"), Note("c'4 * 4/7")]
```

Read *multiplied_durations* cyclically where the length of *multiplied_durations* is less than the length of *pitches*:

```
>>> args = [[0, 2, 4, 5], [(1, 5)]]
>>> notetools.make_quarter_notes_with_lilypond_multipliers(*args)
[Note("c'4 * 4/5"), Note("d'4 * 4/5"), Note("e'4 * 4/5"), Note("f'4 * 4/5")]
```

Return list of zero or more newly constructed notes. Changed in version 2.0: renamed `construct.quarter_notes_with_multipliers()` to `notetools.make_quarter_notes_with_lilypond_multipliers()`.

notetools.make_repeated_notes

`notetools.make_repeated_notes(count, duration=Duration(1, 8))`

Make *count* repeated notes with note head-assignable *duration*:

```
>>> notetools.make_repeated_notes(4)
[Note("c'8"), Note("c'8"), Note("c'8"), Note("c'8")]
```

Make *count* repeated tie chains with tied *duration*:

```
>>> notes = notetools.make_repeated_notes(2, (5, 16))
>>> voice = Voice(notes)
```

```
>>> f(voice)
\new Voice {
  c'4 ~
  c'16
  c'4 ~
  c'16
}
```

Make ad hoc tuplet holding *count* repeated notes with nonbinary *duration*:

```
>>> notetools.make_repeated_notes(3, (1, 12))
[Tuplet(2/3, [c'8, c'8, c'8])]
```

Set pitch of all notes created to middle C.

Return list of zero or more newly constructed notes or list of one newly constructed tuplet. Changed in version 2.0: renamed `construct.run()` to `notetools.make_repeated_notes()`.

notetools.make_repeated_notes_from_time_signature

`notetools.make_repeated_notes_from_time_signature(time_signature, pitch="c")`

New in version 2.0. Make repeated notes from *time_signature*:

```
>>> notetools.make_repeated_notes_from_time_signature((5, 32))
[Note("c'32"), Note("c'32"), Note("c'32"), Note("c'32"), Note("c'32")]
```

Make repeated notes with *pitch* from *time_signature*:

```
>>> notetools.make_repeated_notes_from_time_signature((5, 32), pitch="d'")
[Note("d'32"), Note("d'32"), Note("d'32"), Note("d'32"), Note("d'32")]
```

Return list of notes.

notetools.make_repeated_notes_from_time_signatures

`notetools.make_repeated_notes_from_time_signatures` (*time_signatures*, *pitch*="c")

Make repeated notes from *time_signatures*:

```
notetools.make_repeated_notes_from_time_signatures([(2, 8), (3, 32)])
[[Note("c'8"), Note("c'8")], [Note("c'32"), Note("c'32"), Note("c'32")]]
```

Make repeated notes with *pitch* from *time_signatures*:

```
>>> notetools.make_repeated_notes_from_time_signatures([(2, 8), (3, 32)], pitch="d'")
[[Note("d'8"), Note("d'8")], [Note("d'32"), Note("d'32"), Note("d'32")]]
```

Return two-dimensional list of note lists.

Use `sequencetools.flatten_sequence()` to flatten output if required.

notetools.make_repeated_notes_with_shorter_notes_at_end

`notetools.make_repeated_notes_with_shorter_notes_at_end` (*pitch*, *written_duration*,
total_duration, *prolation*=1)

Make repeated notes with *pitch* and *written_duration* summing to *total_duration* under *prolation*:

```
>>> args = [0, Duration(1, 16), Duration(1, 4)]
>>> notes = notetools.make_repeated_notes_with_shorter_notes_at_end(*args)
>>> voice = Voice(notes)
```

```
>>> f(voice)
\new Voice {
    c'16
    c'16
    c'16
    c'16
}
```

Fill binary remaining duration with binary notes of lesser written duration:

```
>>> args = [0, Duration(1, 16), Duration(9, 32)]
>>> notes = notetools.make_repeated_notes_with_shorter_notes_at_end(*args)
>>> voice = Voice(notes)
```

```
>>> f(voice)
\new Voice {
    c'16
    c'16
    c'16
    c'16
    c'32
}
```

Fill nonbinary remaining duration with ad hoc tuplet:

```
>>> args = [0, Duration(1, 16), Duration(4, 10)]
>>> notes = notetools.make_repeated_notes_with_shorter_notes_at_end(*args)
>>> voice = Voice(notes)
```



```
>>> f(voice)
\new Voice {
  c'16
  c'16
  c'16
  c'16
  c'16
  c'16
  \times 4/5 {
    c'32
  }
}
```

Set *prolation* when constructing notes in a nonbinary measure.

Return list of newly constructed components. Changed in version 2.0: renamed `construct.note_train()` to `notetools.make_repeated_notes_with_shorter_notes_at_end()`.

`notetools.make_tied_note`

`notetools.make_tied_note` (*pitch*, *duration*, *big_endian=True*)

Returns a list of notes to fill the given duration.

Notes returned are tie-spanned.

`notetools.yield_groups_of_notes_in_sequence`

`notetools.yield_groups_of_notes_in_sequence` (*sequence*)

New in version 2.0. Yield groups of notes in *sequence*:

```
>>> staff = Staff("c'8 d'8 r8 r8 <e' g'>8 <f' a'>8 g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

>>> f(staff)
\new Staff {
  c'8
  d'8
  r8
  r8
  <e' g'>8
  <f' a'>8
  g'8
  a'8
  r8
  r8
  <b' d''>8
  <c'' e''>8
}

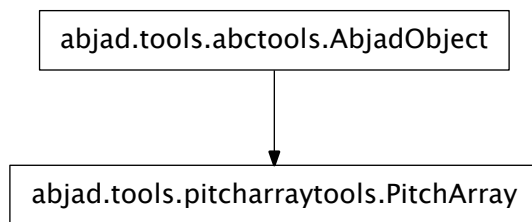
>>> for note in notetools.yield_groups_of_notes_in_sequence(staff):
...     note
...
(Note("c'8"), Note("d'8"))
(Note("g'8"), Note("a'8"))
```

Return generator.

`pitcharraytools`

concrete classes

`pitcharraytools.PitchArray`



class `pitcharraytools.PitchArray` (*args)
New in version 2.0. Two-dimensional array of pitches.

Read-only Properties

`PitchArray.cell_tokens_by_row`

`PitchArray.cell_widths_by_row`

`PitchArray.cells`

`PitchArray.columns`

`PitchArray.depth`

`PitchArray.dimensions`

`PitchArray.has_voice_crossing`

`PitchArray.is_rectangular`

`PitchArray.pitches`

`PitchArray.pitches_by_row`

`PitchArray.rows`

`PitchArray.size`

`PitchArray.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`PitchArray.voice_crossing_count`

`PitchArray.weight`

`PitchArray.width`

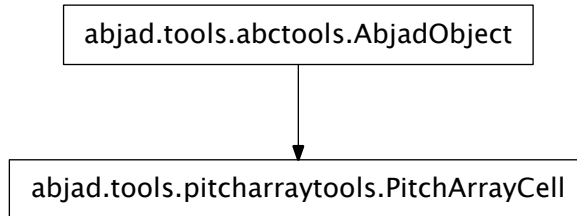
Methods

`PitchArray.append_column(column)`
`PitchArray.append_row(row)`
`PitchArray.apply_pitches_by_row(pitch_lists)`
`PitchArray.copy_subarray(upper_left_pair, lower_right_pair)`
`PitchArray.has_spanning_cell_over_index(index)`
`PitchArray.pad_to_depth(depth)`
`PitchArray.pad_to_width(width)`
`PitchArray.pop_column(column_index)`
`PitchArray.pop_row(row_index=-1)`
`PitchArray.remove_row(row)`

Special Methods

`PitchArray.__add__(arg)`
`PitchArray.__contains__(arg)`
`PitchArray.__eq__(arg)`
`PitchArray.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`
`PitchArray.__getitem__(arg)`
`PitchArray.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`
`PitchArray.__iadd__(arg)`
`PitchArray.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`
`PitchArray.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`
`PitchArray.__ne__(arg)`
`PitchArray.__repr__()`
`PitchArray.__setitem__(i, arg)`
`PitchArray.__str__()`

pitcharraytools.PitchArrayCell



class pitcharraytools.**PitchArrayCell** (*cell_token=None*)

One cell in a pitch array.

```

>>> from abjad.tools import pitcharraytools

>>> array = pitcharraytools.PitchArray([[1, 2, 1], [2, 1, 1]])
>>> print array
[ ] [   ] [ ]
[   ] [ ] [ ]
>>> cell = array[0][1]
>>> cell
PitchArrayCell(x2)

>>> cell.column_indices
(1, 2)

>>> cell.indices
(0, (1, 2))

>>> cell.is_first_in_row
False

>>> cell.is_last_in_row
False

>>> cell.next
PitchArrayCell(x1)

>>> cell.parent_array
PitchArray(PitchArrayRow(x1, x2, x1), PitchArrayRow(x2, x1, x1))

>>> cell.parent_column
PitchArrayColumn(x2, x2)

>>> cell.parent_row
PitchArrayRow(x1, x2, x1)

>>> cell.pitches
[]
  
```

```
>>> cell.prev
PitchArrayCell(x1)

>>> cell.row_index
0

>>> cell.token
2

>>> cell.width
2
```

Return pitch array cell.

Read-only Properties

`PitchArrayCell.column_indices`

Read-only tuple of one or more nonnegative integer indices.

`PitchArrayCell.indices`

`PitchArrayCell.is_first_in_row`

`PitchArrayCell.is_last_in_row`

`PitchArrayCell.next`

`PitchArrayCell.parent_array`

`PitchArrayCell.parent_column`

`PitchArrayCell.parent_row`

`PitchArrayCell.prev`

`PitchArrayCell.row_index`

`PitchArrayCell.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`PitchArrayCell.token`

`PitchArrayCell.weight`

`PitchArrayCell.width`

Read/write Properties

`PitchArrayCell.pitches`

Methods

`PitchArrayCell.matches_cell` (*arg*)

Special Methods

`PitchArrayCell.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`PitchArrayCell.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PitchArrayCell.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`PitchArrayCell.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PitchArrayCell.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PitchArrayCell.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

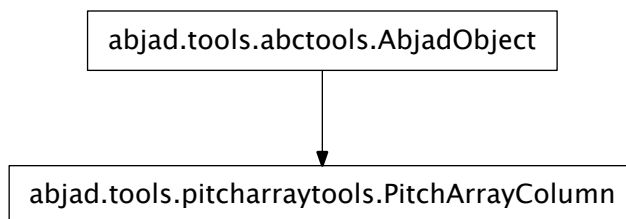
Return boolean.

Inherited from `abctools.AbjadObject`

`PitchArrayCell.__repr__()`

`PitchArrayCell.__str__()`

pitcharraytools.PitchArrayColumn



class `pitcharraytools.PitchArrayColumn` (*cells*)

New in version 2.0. Column in a pitch array:

```
>>> from abjad.tools import pitcharraytools
```

```

>>> array = pitcharraytools.PitchArray([
...     [1, (2, 1), (-1.5, 2)],
...     [(7, 2), (6, 1), 1]])

>>> print array
[ ] [d'] [bqf  ]
[g'    ] [fs'] [ ]

>>> array.columns[0]
PitchArrayColumn(x1, g' x2)

>>> print array.columns[0]
[ ]
[g'    ]

```

Return pitch array column.

Read-only Properties

PitchArrayColumn.cell_tokens
PitchArrayColumn.cell_widths
PitchArrayColumn.cells
PitchArrayColumn.column_index
PitchArrayColumn.depth
PitchArrayColumn.dimensions
PitchArrayColumn.has_voice_crossing
PitchArrayColumn.is_defective
PitchArrayColumn.parent_array
PitchArrayColumn.pitches
PitchArrayColumn.start_cells
PitchArrayColumn.start_pitches
PitchArrayColumn.stop_cells
PitchArrayColumn.stop_pitches
PitchArrayColumn.storage_format
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

PitchArrayColumn.weight
PitchArrayColumn.width

Methods

PitchArrayColumn.append(*cell*)
PitchArrayColumn.extend(*cells*)
PitchArrayColumn.remove_pitches()

Special Methods

`PitchArrayColumn.__eq__(arg)`

`PitchArrayColumn.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PitchArrayColumn.__getitem__(arg)`

`PitchArrayColumn.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`PitchArrayColumn.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PitchArrayColumn.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

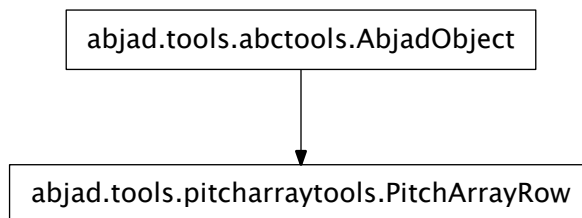
Inherited from `abctools.AbjadObject`

`PitchArrayColumn.__ne__(arg)`

`PitchArrayColumn.__repr__()`

`PitchArrayColumn.__str__()`

`pitcharraytools.PitchArrayRow`



class `pitcharraytools.PitchArrayRow(cells)`

New in version 2.0. One row in pitch array.

```
>>> from abjad.tools import pitcharraytools
```



```

>>> array = pitcharraytools.PitchArray([[1, 2, 1], [2, 1, 1]])
>>> array[0].cells[0].pitches.append(0)
>>> array[0].cells[1].pitches.append(2)
>>> array[1].cells[2].pitches.append(4)
>>> print array
[c'] [d'   ] [  ]
[    ] [  ] [e']

>>> array[0]
PitchArrayRow(c', d' x2, x1)

>>> array[0].cell_widths
(1, 2, 1)

>>> array[0].dimensions
(1, 4)

>>> array[0].pitches
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"))

```

Return pitch array row.

Read-only Properties

`PitchArrayRow.cell_tokens`

`PitchArrayRow.cell_widths`

`PitchArrayRow.cells`

`PitchArrayRow.depth`

`PitchArrayRow.dimensions`

`PitchArrayRow.is_defective`

`PitchArrayRow.is_in_range`

`PitchArrayRow.parent_array`

`PitchArrayRow.pitches`

`PitchArrayRow.row_index`

`PitchArrayRow.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`PitchArrayRow.weight`

`PitchArrayRow.width`

Read/write Properties

`PitchArrayRow.pitch_range`

Methods

`PitchArrayRow.append(cell_token)`

`PitchArrayRow.apply_pitches(pitch_tokens)`

`PitchArrayRow.copy_subrow(start=None, stop=None)`
`PitchArrayRow.empty_pitches()`
`PitchArrayRow.extend(cell_tokens)`
`PitchArrayRow.has_spanning_cell_over_index(i)`
`PitchArrayRow.index(cell)`
`PitchArrayRow.merge(cells)`
`PitchArrayRow.pad_to_width(width)`
`PitchArrayRow.pop(cell_index)`
`PitchArrayRow.remove(cell)`
`PitchArrayRow.withdraw()`

Special Methods

`PitchArrayRow.__add__(arg)`
`PitchArrayRow.__eq__(arg)`
`PitchArrayRow.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`
`PitchArrayRow.__getitem__(arg)`
`PitchArrayRow.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`
`PitchArrayRow.__iadd__(arg)`
`PitchArrayRow.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`
`PitchArrayRow.__len__()`
`PitchArrayRow.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`
`PitchArrayRow.__ne__(arg)`
`PitchArrayRow.__repr__()`
`PitchArrayRow.__str__()`

functions

pitcharraytools.all_are_pitch_arrays

`pitcharraytools.all_are_pitch_arrays(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad pitch arrays:

```
>>> pitch_array = pitcharraytools.PitchArray([[1, 2, 1], [2, 1, 1]])
```

```
>>> pitcharraytools.all_are_pitch_arrays([pitch_array])
True
```

True when *expr* is an empty sequence:

```
>>> pitcharraytools.all_are_pitch_arrays([])
True
```

Otherwise false:

```
>>> pitcharraytools.all_are_pitch_arrays('foo')
False
```

Return boolean.

pitcharraytools.concatenate_pitch_arrays

`pitcharraytools.concatenate_pitch_arrays(pitch_arrays)`

New in version 2.0. Concatenate *pitch_arrays*:

```
>>> array_1 = pitcharraytools.PitchArray([[1, 2, 1], [2, 1, 1]])
>>> print array_1
[ ] [ ] [ ]
[ ] [ ] [ ]
```

```
>>> array_2 = pitcharraytools.PitchArray([[3, 4], [4, 3]])
>>> print array_2
[ ] [ ]
[ ] [ ]
```

```
>>> array_3 = pitcharraytools.PitchArray([[1, 1], [1, 1]])
>>> print array_3
[ ] [ ]
[ ] [ ]
```

```
>>> merged_array = pitcharraytools.concatenate_pitch_arrays([array_1, array_2, array_3])
>>> print merged_array
[ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ]
```

Return pitch array.

pitcharraytools.list_nonspanning_subarrays_of_pitch_array

`pitcharraytools.list_nonspanning_subarrays_of_pitch_array(pitch_array)`

New in version 2.0. List nonspanning subarrays of *pitch_array*:

```
>>> array = pitcharraytools.PitchArray([
...     [2, 2, 3, 1],
...     [1, 2, 1, 1, 2, 1],
...     [1, 1, 1, 1, 1, 1, 1, 1]])
>>> print array
[ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ]

>>> subarrays = pitcharraytools.list_nonspanning_subarrays_of_pitch_array(array)
>>> len(subarrays)
3

>>> print subarrays[0]
[ ] [ ]
[ ] [ ] [ ]
[ ] [ ] [ ] [ ]

>>> print subarrays[1]
[ ]
[ ] [ ]
[ ] [ ] [ ]

>>> print subarrays[2]
[ ]
[ ]
[ ]
```

Return list.

`pitcharraytools.make_empty_pitch_array_from_list_of_pitch_lists`

`pitcharraytools.make_empty_pitch_array_from_list_of_pitch_lists` (*leaf_iterables*)

New in version 2.0. Make empty pitch array from *leaf_iterables*:

```
>>> score = Score([])
>>> score.append(Staff("c'8 d'8 e'8 f'8"))
>>> score.append(Staff("c'4 d'4"))
>>> score.append(Staff(tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8") * 2))

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
  \new Staff {
    c'4
    d'4
  }
  \new Staff {
    \times 2/3 {
      c'8
      d'8
      e'8
    }
  }
}
```

```

    }
    \times 2/3 {
      c'8
      d'8
      e'8
    }
  }
}

>>

>>> array = pitcharraytools.make_empty_pitch_array_from_list_of_pitch_lists(score)
>>> print array
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

```

Return pitch array.

`pitcharraytools.make_pitch_array_score_from_pitch_arrays`

`pitcharraytools.make_pitch_array_score_from_pitch_arrays` (*pitch_arrays*)

New in version 2.0. Make pitch-array score from *pitch_arrays*:

```

>>> from abjad.tools import pitcharraytools

>>> array_1 = pitcharraytools.PitchArray([
...     [1, (2, 1), ([-2, -1.5], 2)],
...     [(7, 2), (6, 1), 1]])

>>> array_2 = pitcharraytools.PitchArray([
...     [1, 1, 1],
...     [1, 1, 1]])

>>> score = pitcharraytools.make_pitch_array_score_from_pitch_arrays([array_1, array_2])

>>> f(score)
\new Score <<
  \new StaffGroup <<
    \new Staff {
      {
        \time 4/8
        r8
        d'8
        <bf bqf>4
      }
      {
        \time 3/8
        r8
        r8
        r8
      }
    }
  \new Staff {
    {
      \time 4/8
      g'4
      fs'8
      r8
    }
  }
}

```

```
        }
        {
            \time 3/8
            r8
            r8
            r8
        }
    }
>>
>>
```

Create one staff per pitch-array row.

Return score.

pitcharraytools.make_populated_pitch_array_from_list_of_pitch_lists

`pitcharraytools.make_populated_pitch_array_from_list_of_pitch_lists` (*leaf_iterables*)

New in version 2.0. Make populated pitch array from *leaf_iterables*:

```
>>> from abjad.tools import pitcharraytools

>>> score = Score([])
>>> score.append(Staff("c'8 d'8 e'8 f'8"))
>>> score.append(Staff("c'4 d'4"))
>>> score.append(Staff(tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8") * 2))
>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
  \new Staff {
    c'4
    d'4
  }
  \new Staff {
    \times 2/3 {
      c'8
      d'8
      e'8
    }
    \times 2/3 {
      c'8
      d'8
      e'8
    }
  }
>>

>>> array = pitcharraytools.make_populated_pitch_array_from_list_of_pitch_lists(score)
>>> print array
[c'      ] [d'      ] [e'      ] [f'      ]
[c'      ] [d'      ] [d'      ]
[c'] [d'      ] [e'] [c'] [d'      ] [e']
```

Return pitch array.

`pitcharraytools.pitch_array_row_to_measure`

`pitcharraytools.pitch_array_row_to_measure` (*pitch_array_row*,

cell_duration_denominator=8)

New in version 2.0. Change *pitch_array_row* to measure with meter *pitch_array_row.width* over *cell_duration_denominator*:

```
>>> from abjad.tools import pitcharraytools
>>> array = pitcharraytools.PitchArray([
...     [1, (2, 1), ([-2, -1.5], 2)],
...     [(7, 2), (6, 1), 1]])

>>> print array
[ ] [d'] [bf bqf  ]
[g'   ] [fs'   ] [ ]

>>> measure = pitcharraytools.pitch_array_row_to_measure(array.rows[0])

>>> f(measure)
{
    \time 4/8
    r8
    d'8
    <bf bqf>4
}
```

Return measure.

`pitcharraytools.pitch_array_to_measures`

`pitcharraytools.pitch_array_to_measures` (*pitch_array*, *cell_duration_denominator=8*)

New in version 2.0. Change *pitch_array* to measures with meters *row.width* over *cell_duration_denominator* for each row in *pitch_array*:

```
>>> from abjad.tools import pitcharraytools
>>> array = pitcharraytools.PitchArray([
...     [1, (2, 1), ([-2, -1.5], 2)],
...     [(7, 2), (6, 1), 1]])

>>> print array
[ ] [d'] [bf bqf  ]
[g'   ] [fs'   ] [ ]

>>> pitcharraytools.pitch_array_to_measures(array)
[Measure(4/8, [r8, d'8, <bf bqf>4]), Measure(4/8, [g'4, fs'8, r8])]
>>> for measure in _:
...     f(measure)
...
{
    \time 4/8
    r8
    d'8
    <bf bqf>4
}
```

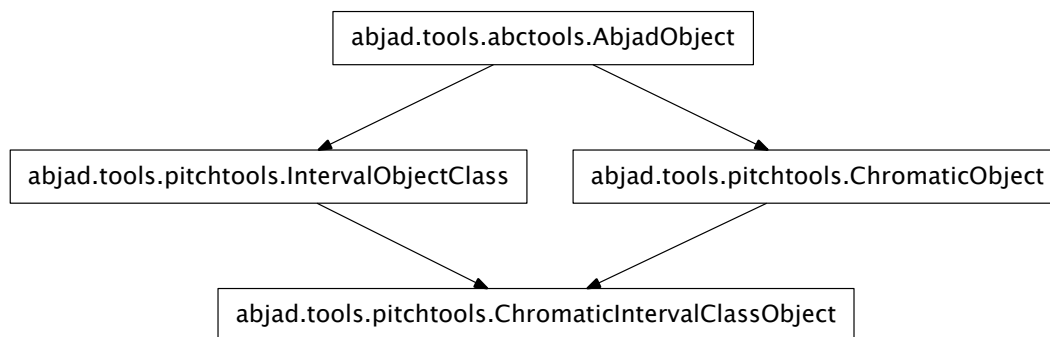
```
{
    \time 4/8
    g' 4
    fs' 8
    r8
}
```

Return list of measures.

pitchtools

abstract classes

pitchtools.ChromaticIntervalClassObject



class `pitchtools.ChromaticIntervalClassObject`

New in version 2.0. Chromatic interval-class base class.

Read-only Properties

`ChromaticIntervalClassObject.number`

Inherited from `pitchtools.IntervalObjectClass`

`ChromaticIntervalClassObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ChromaticIntervalClassObject.__abs__()`

`ChromaticIntervalClassObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ChromaticIntervalClassObject.__float__()`
`ChromaticIntervalClassObject.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`ChromaticIntervalClassObject.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`ChromaticIntervalClassObject.__hash__()`
Inherited from `pitchtools.IntervalObjectClass`

`ChromaticIntervalClassObject.__int__()`

`ChromaticIntervalClassObject.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

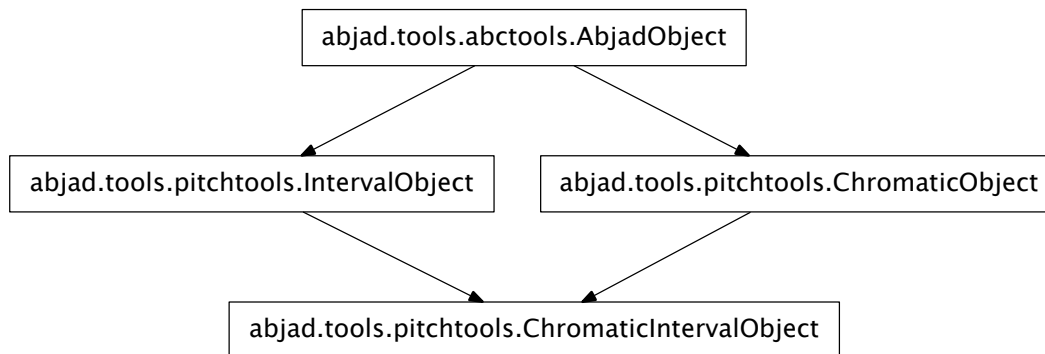
`ChromaticIntervalClassObject.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`ChromaticIntervalClassObject.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`ChromaticIntervalClassObject.__repr__()`
Inherited from `pitchtools.IntervalObjectClass`

`ChromaticIntervalClassObject.__str__()`
Inherited from `pitchtools.IntervalObjectClass`

pitchtools.ChromaticIntervalObject



class `pitchtools.ChromaticIntervalObject` (*arg*)
 New in version 2.0. Chromatic interval base class.

Read-only Properties

`ChromaticIntervalObject.cents`
 Inherited from `pitchtools.IntervalObject`

`ChromaticIntervalObject.interval_class`
 Inherited from `pitchtools.IntervalObject`

`ChromaticIntervalObject.number`

`ChromaticIntervalObject.semitones`

`ChromaticIntervalObject.storage_format`
 Storage format of Abjad object.
 Return string.
 Inherited from `abctools.AbjadObject`

Special Methods

`ChromaticIntervalObject.__abs__()`

`ChromaticIntervalObject.__add__(arg)`

`ChromaticIntervalObject.__eq__(arg)`

`ChromaticIntervalObject.__float__()`

`ChromaticIntervalObject.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`ChromaticIntervalObject.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Inherited from `abctools.AbjadObject`

`ChromaticIntervalObject.__hash__()`

Inherited from `pitchtools.IntervalObject`

`ChromaticIntervalObject.__int__()`

`ChromaticIntervalObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ChromaticIntervalObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

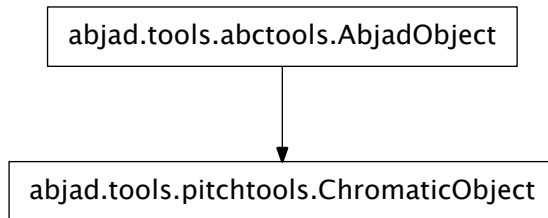
`ChromaticIntervalObject.__ne__(arg)`

`ChromaticIntervalObject.__repr__()`

`ChromaticIntervalObject.__str__()`

`ChromaticIntervalObject.__sub__(arg)`

`pitchtools.ChromaticObject`



class `pitchtools.ChromaticObject`

`..versionadded:: 2.0`

Chromatic object base class.

Read-only Properties

`ChromaticObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ChromaticObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ChromaticObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ChromaticObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ChromaticObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ChromaticObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ChromaticObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

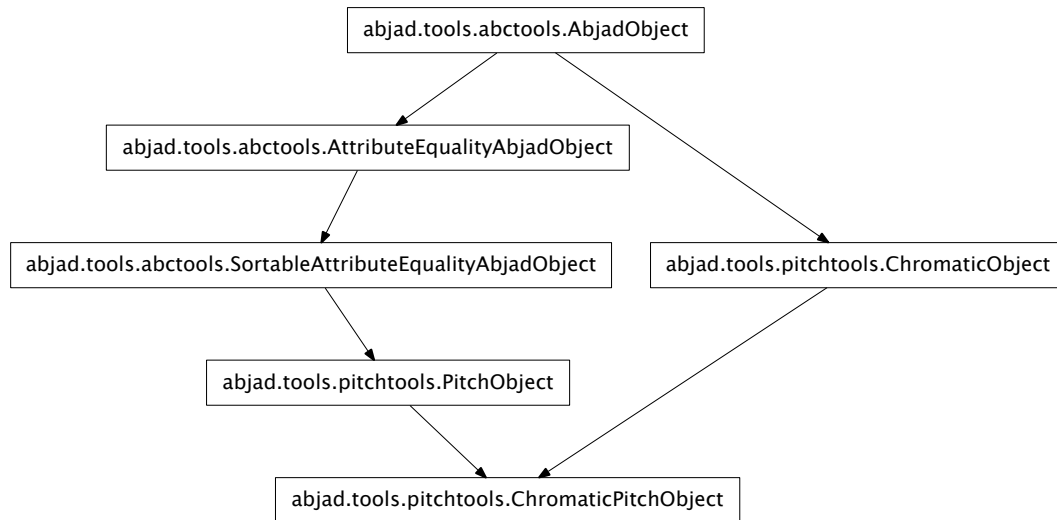
`ChromaticObject.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

pitchtools.ChromaticPitchObject



class `pitchtools.ChromaticPitchObject`
 New in version 2.0. Chromatic pitch base class.

Read-only Properties

`ChromaticPitchObject.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ChromaticPitchObject.__abs__()`
 Inherited from `pitchtools.PitchObject`

`ChromaticPitchObject.__eq__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`ChromaticPitchObject.__float__()`
 Inherited from `pitchtools.PitchObject`

`ChromaticPitchObject.__ge__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`ChromaticPitchObject.__gt__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.
 Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`ChromaticPitchObject.__hash__()`
 Inherited from `pitchtools.PitchObject`

`ChromaticPitchObject.__int__()`
 Inherited from `pitchtools.PitchObject`

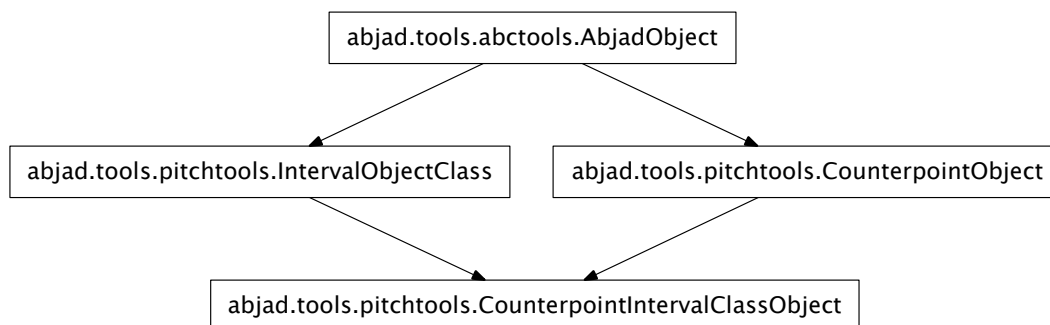
`ChromaticPitchObject.__le__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.
 Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`ChromaticPitchObject.__lt__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.
 Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`ChromaticPitchObject.__ne__(arg)`
 Inherited from `pitchtools.PitchObject`

`ChromaticPitchObject.__repr__()`
 Inherited from `pitchtools.PitchObject`

`pitchtools.CounterpointIntervalClassObject`



`class pitchtools.CounterpointIntervalClassObject`
 New in version 2.0. Counterpoint interval-class base class.

Read-only Properties

`CounterpointIntervalClassObject.number`
 Inherited from `pitchtools.IntervalObjectClass`

`CounterpointIntervalClassObject.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`CounterpointIntervalClassObject.__abs__()`

`CounterpointIntervalClassObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`CounterpointIntervalClassObject.__float__()`

`CounterpointIntervalClassObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CounterpointIntervalClassObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`CounterpointIntervalClassObject.__hash__()`

Inherited from `pitchtools.IntervalObjectClass`

`CounterpointIntervalClassObject.__int__()`

`CounterpointIntervalClassObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CounterpointIntervalClassObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CounterpointIntervalClassObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

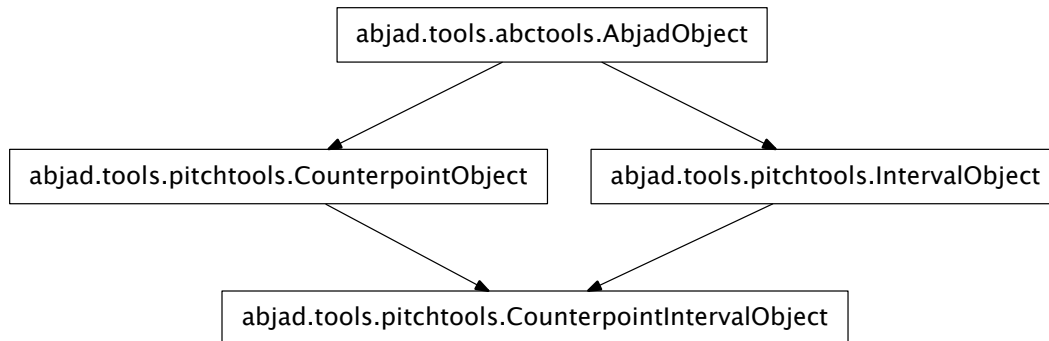
`CounterpointIntervalClassObject.__repr__()`

Inherited from `pitchtools.IntervalObjectClass`

`CounterpointIntervalClassObject.__str__()`

Inherited from `pitchtools.IntervalObjectClass`

pitchtools.CounterpointIntervalObject



```
class pitchtools.CounterpointIntervalObject
    ..versionadded:: 2.0

    Counterpoint interval base class.
```

Read-only Properties

```
CounterpointIntervalObject.cents
    Inherited from pitchtools.IntervalObject
CounterpointIntervalObject.interval_class
    Inherited from pitchtools.IntervalObject
CounterpointIntervalObject.number
CounterpointIntervalObject.semitones
CounterpointIntervalObject.storage_format
    Storage format of Abjad object.

    Return string.

    Inherited from abctools.AbjadObject
```

Special Methods

```
CounterpointIntervalObject.__abs__()
CounterpointIntervalObject.__eq__(arg)
    True when id(self) equals id(arg).

    Return boolean.

    Inherited from abctools.AbjadObject
CounterpointIntervalObject.__float__()
CounterpointIntervalObject.__ge__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject
```


`CounterpointIntervalObject.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`CounterpointIntervalObject.__hash__()`
 Inherited from `pitchtools.IntervalObject`

`CounterpointIntervalObject.__int__()`

`CounterpointIntervalObject.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

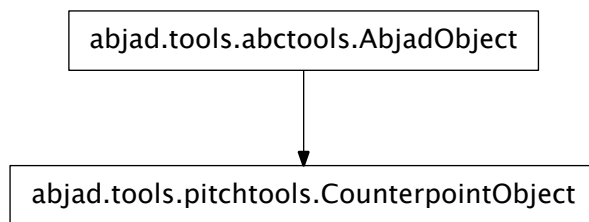
`CounterpointIntervalObject.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`CounterpointIntervalObject.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`CounterpointIntervalObject.__repr__()`
 Inherited from `pitchtools.IntervalObject`

`CounterpointIntervalObject.__str__()`
 Inherited from `pitchtools.IntervalObject`

pitchtools.CounterpointObject



class `pitchtools.CounterpointObject`
 Counterpoint object base class.

Read-only Properties

`CounterpointObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`CounterpointObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`CounterpointObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CounterpointObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`CounterpointObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CounterpointObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CounterpointObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

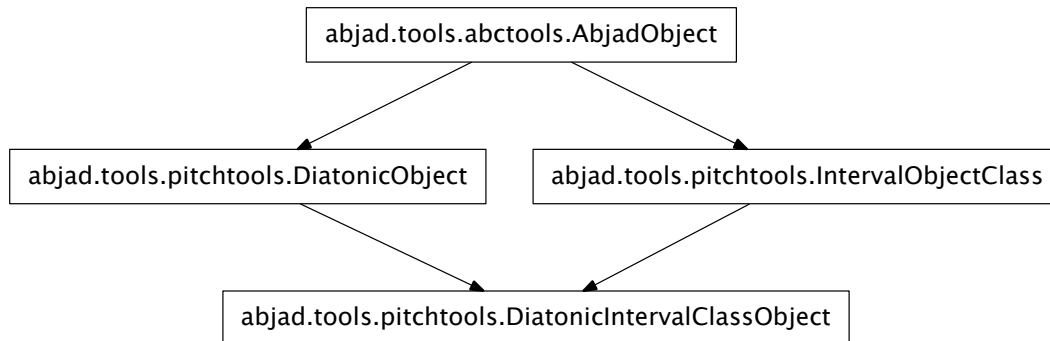
`CounterpointObject.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

pitchtools.DiatonicIntervalClassObject



class `pitchtools.DiatonicIntervalClassObject`
 New in version 2.0. Diatonic interval-class base class.

Read-only Properties

`DiatonicIntervalClassObject.number`
 Inherited from `pitchtools.IntervalObjectClass`

`DiatonicIntervalClassObject.quality_string`
`DiatonicIntervalClassObject.storage_format`
 Storage format of Abjad object.
 Return string.
 Inherited from `abctools.AbjadObject`

Special Methods

`DiatonicIntervalClassObject.__abs__()`
`DiatonicIntervalClassObject.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`DiatonicIntervalClassObject.__float__()`
`DiatonicIntervalClassObject.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`DiatonicIntervalClassObject.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`DiatonicIntervalClassObject.__hash__()`
 Inherited from `pitchtools.IntervalObjectClass`

`DiatonicIntervalClassObject.__int__()`

`DiatonicIntervalClassObject.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DiatonicIntervalClassObject.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DiatonicIntervalClassObject.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

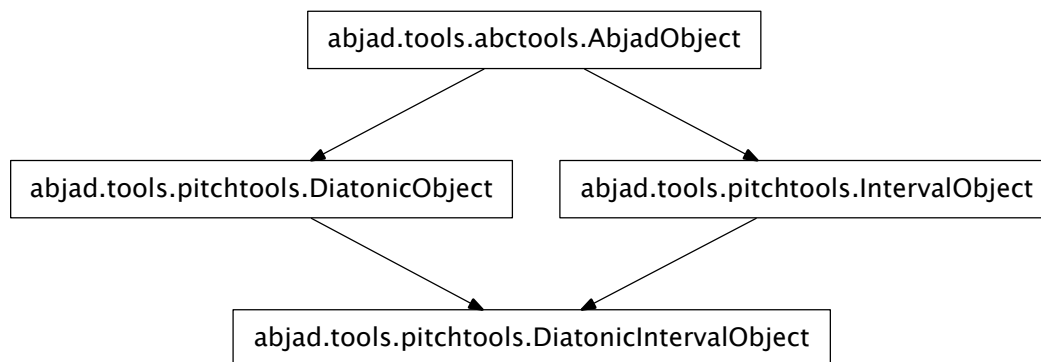
Return boolean.

Inherited from `abctools.AbjadObject`

`DiatonicIntervalClassObject.__repr__()`

`DiatonicIntervalClassObject.__str__()`

`pitchtools.DiatonicIntervalObject`



class `pitchtools.DiatonicIntervalObject` (*quality_string, number*)
 New in version 2.0. Diatonic interval base class.

Read-only Properties

`DiatonicIntervalObject.cents`

Inherited from `pitchtools.IntervalObject`

`DiatonicIntervalObject.diatonic_interval_class`

`DiatonicIntervalObject.interval_class`

DiatonicIntervalObject.**interval_string**

DiatonicIntervalObject.**number**

DiatonicIntervalObject.**quality_string**

DiatonicIntervalObject.**semitones**

DiatonicIntervalObject.**staff_spaces**

DiatonicIntervalObject.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

DiatonicIntervalObject.**__abs__**()

DiatonicIntervalObject.**__eq__**(arg)

DiatonicIntervalObject.**__float__**()

DiatonicIntervalObject.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

DiatonicIntervalObject.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

DiatonicIntervalObject.**__hash__**()

Inherited from `pitchtools.IntervalObject`

DiatonicIntervalObject.**__int__**()

DiatonicIntervalObject.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

DiatonicIntervalObject.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

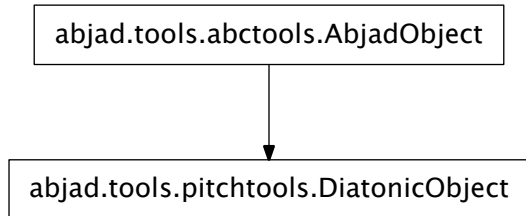
Inherited from `abctools.AbjadObject`

DiatonicIntervalObject.**__ne__**(arg)

DiatonicIntervalObject.**__repr__**()

DiatonicIntervalObject.**__str__**()

pitchtools.DiatonicObject



class `pitchtools.DiatonicObject`

..versionadded:: 2.0

Diatonic object base class.

Read-only Properties

`DiatonicObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`DiatonicObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DiatonicObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DiatonicObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DiatonicObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DiatonicObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DiatonicObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

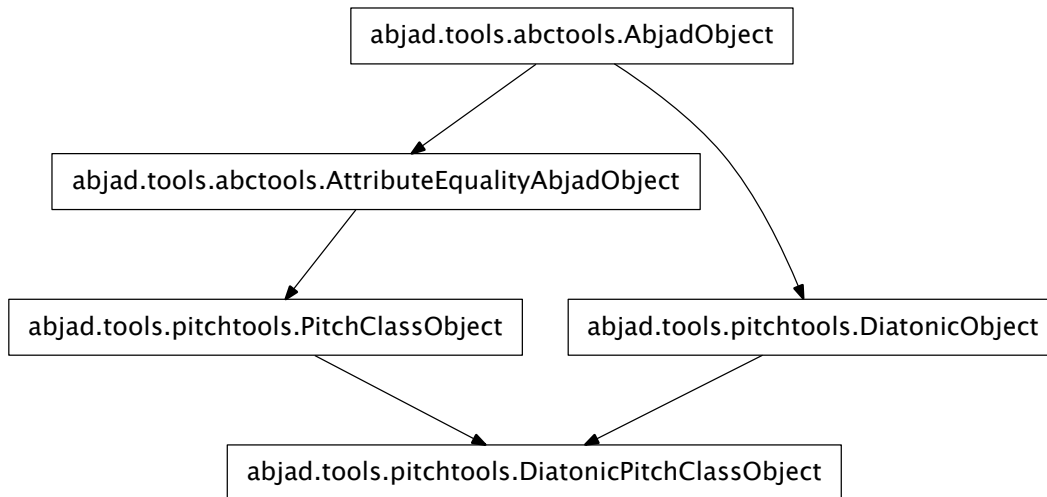
`DiatonicObject.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`pitchtools.DiatonicPitchClassObject`



`class pitchtools.DiatonicPitchClassObject`

New in version 2.0. Diatonic pitch-class base class.

Read-only Properties

`DiatonicPitchClassObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`DiatonicPitchClassObject.__abs__()`

`DiatonicPitchClassObject.__eq__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.
Inherited from `abctools.AttributeEqualityAbjadObject`

`DiatonicPitchClassObject.__float__()`

`DiatonicPitchClassObject.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`DiatonicPitchClassObject.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`DiatonicPitchClassObject.__hash__()`
Inherited from `pitchtools.PitchClassObject`

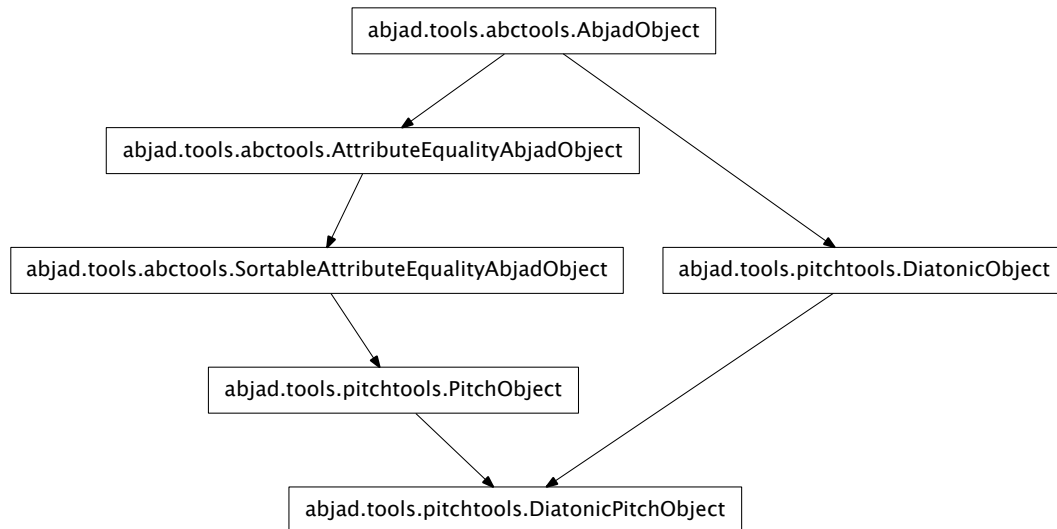
`DiatonicPitchClassObject.__int__()`

`DiatonicPitchClassObject.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`DiatonicPitchClassObject.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`DiatonicPitchClassObject.__ne__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.
Inherited from `abctools.AttributeEqualityAbjadObject`

`DiatonicPitchClassObject.__repr__()`
Interpreter representation of object defined equal to class name and format string.
Return string.
Inherited from `abctools.AttributeEqualityAbjadObject`

pitchtools.DiatonicPitchObject

class `pitchtools.DiatonicPitchObject`
 New in version 2.0. Diatonic pitch base class.

Read-only Properties

`DiatonicPitchObject.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`DiatonicPitchObject.__abs__()`

`DiatonicPitchObject.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`DiatonicPitchObject.__float__()`

`DiatonicPitchObject.__ge__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`DiatonicPitchObject.__gt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`DiatonicPitchObject.__hash__()`

Inherited from `pitchtools.PitchObject`

`DiatonicPitchObject.__int__()`

`DiatonicPitchObject.__le__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`DiatonicPitchObject.__lt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

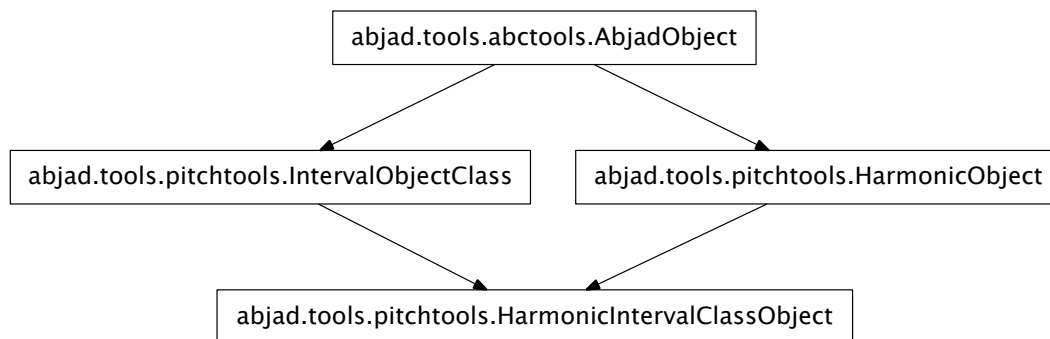
`DiatonicPitchObject.__ne__(arg)`

Inherited from `pitchtools.PitchObject`

`DiatonicPitchObject.__repr__()`

Inherited from `pitchtools.PitchObject`

`pitchtools.HarmonicIntervalClassObject`



`class pitchtools.HarmonicIntervalClassObject`

New in version 2.0. Harmonic interval-class base class.

Read-only Properties

`HarmonicIntervalClassObject.number`

Inherited from `pitchtools.IntervalObjectClass`

`HarmonicIntervalClassObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`HarmonicIntervalClassObject.__abs__()`

Inherited from `pitchtools.IntervalObjectClass`

`HarmonicIntervalClassObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`HarmonicIntervalClassObject.__float__()`

Inherited from `pitchtools.IntervalObjectClass`

`HarmonicIntervalClassObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HarmonicIntervalClassObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`HarmonicIntervalClassObject.__hash__()`

Inherited from `pitchtools.IntervalObjectClass`

`HarmonicIntervalClassObject.__int__()`

Inherited from `pitchtools.IntervalObjectClass`

`HarmonicIntervalClassObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HarmonicIntervalClassObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HarmonicIntervalClassObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

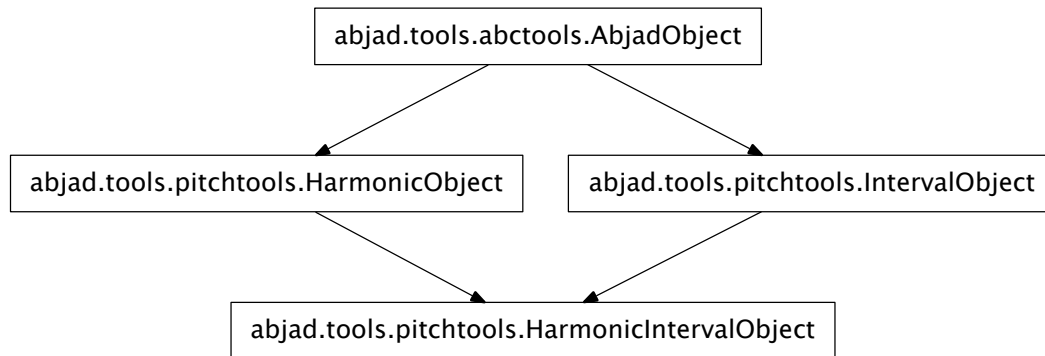
`HarmonicIntervalClassObject.__repr__()`

Inherited from `pitchtools.IntervalObjectClass`

`HarmonicIntervalClassObject.__str__()`

Inherited from `pitchtools.IntervalObjectClass`

pitchtools.HarmonicIntervalObject



```
class pitchtools.HarmonicIntervalObject
    ..versionadded:: 2.0
```

Harmonic interval base class.

Read-only Properties

```
HarmonicIntervalObject.cents
    Inherited from pitchtools.IntervalObject
HarmonicIntervalObject.interval_class
    Inherited from pitchtools.IntervalObject
HarmonicIntervalObject.number
    Inherited from pitchtools.IntervalObject
HarmonicIntervalObject.semitones
    Inherited from pitchtools.IntervalObject
HarmonicIntervalObject.storage_format
    Storage format of Abjad object.

    Return string.

    Inherited from abctools.AbjadObject
```

Special Methods

```
HarmonicIntervalObject.__abs__()
    Inherited from pitchtools.IntervalObject
HarmonicIntervalObject.__eq__(arg)
    True when id(self) equals id(arg).

    Return boolean.

    Inherited from abctools.AbjadObject
HarmonicIntervalObject.__float__()
    Inherited from pitchtools.IntervalObject
```

`HarmonicIntervalObject.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`HarmonicIntervalObject.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`HarmonicIntervalObject.__hash__()`
 Inherited from `pitchtools.IntervalObject`

`HarmonicIntervalObject.__int__()`
 Inherited from `pitchtools.IntervalObject`

`HarmonicIntervalObject.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

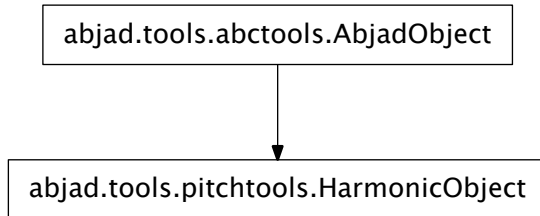
`HarmonicIntervalObject.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`HarmonicIntervalObject.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`HarmonicIntervalObject.__repr__()`
 Inherited from `pitchtools.IntervalObject`

`HarmonicIntervalObject.__str__()`
 Inherited from `pitchtools.IntervalObject`

pitchtools.HarmonicObject



class `pitchtools.HarmonicObject`

..versionadded:: 2.0

Harmonic object base class.

Read-only Properties

`HarmonicObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`HarmonicObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`HarmonicObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HarmonicObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`HarmonicObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HarmonicObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HarmonicObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

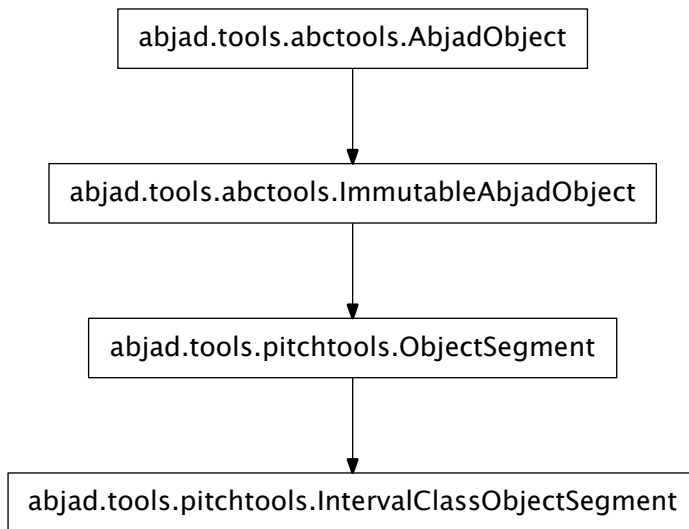
`HarmonicObject.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`pitchtools.IntervalClassObjectSegment`



class `pitchtools.IntervalClassObjectSegment` (*args, **kwargs)

New in version 2.0. Interval-class segment base class.

Read-only Properties

`IntervalClassObjectSegment.interval_class_numbers`

`IntervalClassObjectSegment.interval_classes`

`IntervalClassObjectSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

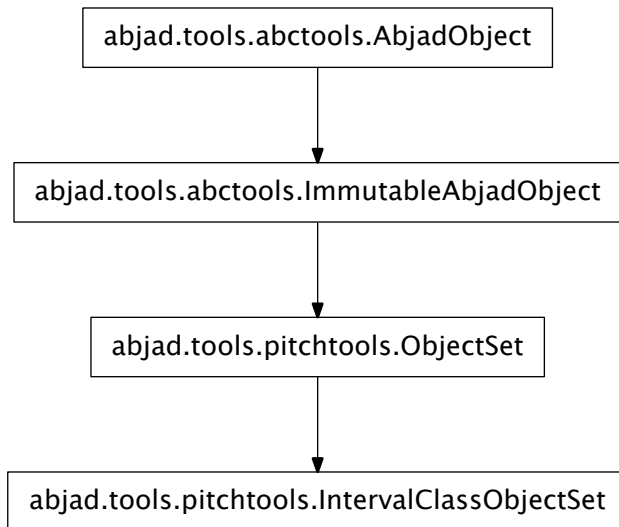
`IntervalClassObjectSegment.count(value) → integer` – return number of occurrences of value
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.index(value[, start[, stop]]) → integer` – return first index of value.
 Raises `ValueError` if the value is not present.
 Inherited from `__builtin__.tuple`

Special Methods

`IntervalClassObjectSegment.__add__(arg)`
 Inherited from `pitchtools.ObjectSegment`
`IntervalClassObjectSegment.__contains__()`
`x.__contains__(y) <==> y in x`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__eq__()`
`x.__eq__(y) <==> x==y`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__ge__()`
`x.__ge__(y) <==> x>=y`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__getitem__()`
`x.__getitem__(y) <==> x[y]`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__getslice__(start, stop)`
 Inherited from `pitchtools.ObjectSegment`
`IntervalClassObjectSegment.__gt__()`
`x.__gt__(y) <==> x>y`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__hash__()` <==> `hash(x)`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__iter__()` <==> `iter(x)`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__le__()`
`x.__le__(y) <==> x<=y`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__len__()` <==> `len(x)`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__lt__()`
`x.__lt__(y) <==> x<y`
 Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__mul__(n)`
 Inherited from `pitchtools.ObjectSegment`
`IntervalClassObjectSegment.__ne__()`
`x.__ne__(y) <==> x!=y`

Inherited from `__builtin__.tuple`
`IntervalClassObjectSegment.__repr__()`
`IntervalClassObjectSegment.__rmul__(n)`
 Inherited from `pitchtools.ObjectSegment`

`pitchtools.IntervalClassObjectSet`



`class pitchtools.IntervalClassObjectSet(*args, **kwargs)`
 New in version 2.0. Interval-class set base class.

Read-only Properties

`IntervalClassObjectSet.storage_format`
 Storage format of Abjad object.
 Return string.
 Inherited from `abctools.AbjadObject`

Methods

`IntervalClassObjectSet.copy()`
 Return a shallow copy of a set.
 Inherited from `__builtin__.frozenset`
`IntervalClassObjectSet.difference()`
 Return the difference of two or more sets as a new set.
 (i.e. all elements that are in this set but not the others.)
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.intersection()`
 Return the intersection of two or more sets as a new set.
 (i.e. elements that are common to all of the sets.)
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.isdisjoint()`
 Return True if two sets have a null intersection.
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.issubset()`
 Report whether another set contains this set.
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.issuperset()`
 Report whether this set contains another set.
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.symmetric_difference()`
 Return the symmetric difference of two sets as a new set.
 (i.e. all elements that are in exactly one of the sets.)
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.union()`
 Return the union of sets as a new set.
 (i.e. all elements that are in either set.)
 Inherited from `__builtin__.frozenset`

Special Methods

`IntervalClassObjectSet.__and__()`
`x.__and__(y) <==> x&y`
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.__cmp__(y) <==> cmp(x, y)`
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.__contains__()`
`x.__contains__(y) <==> y in x.`
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.__eq__()`
`x.__eq__(y) <==> x==y`
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.__ge__()`
`x.__ge__(y) <==> x>=y`
 Inherited from `__builtin__.frozenset`

`IntervalClassObjectSet.__gt__()`
`x.__gt__(y) <==> x>y`
 Inherited from `__builtin__.frozenset`

```

IntervalClassObjectSet.__hash__() <==> hash(x)
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__iter__() <==> iter(x)
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__repr__() <==> repr(x)
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

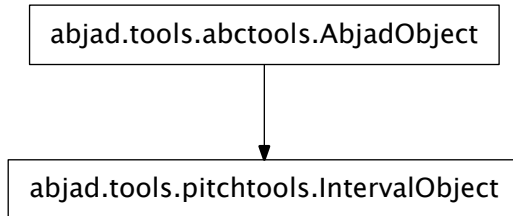
IntervalClassObjectSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

IntervalClassObjectSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset

```

pitchtools.IntervalObject



class `pitchtools.IntervalObject`
 New in version 2.0. Interval base class.

Read-only Properties

`IntervalObject.cents`

`IntervalObject.interval_class`

`IntervalObject.number`

`IntervalObject.semitones`

`IntervalObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`IntervalObject.__abs__()`

`IntervalObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`IntervalObject.__float__()`

`IntervalObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`IntervalObject.__gt__(arg)`

Abjad objects by default do not implement this method.

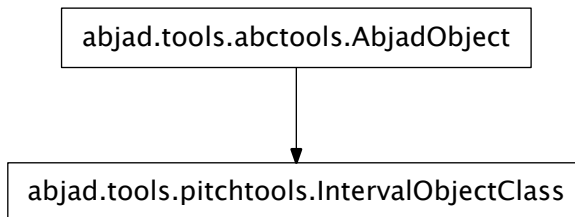
Raise exception

Inherited from `abctools.AbjadObject`

`IntervalObject.__hash__()`

```
IntervalObject.__int__()
IntervalObject.__le__(arg)
    Abjad objects by default do not implement this method.
    Raise exception.
    Inherited from abctools.AbjadObject
IntervalObject.__lt__(arg)
    Abjad objects by default do not implement this method.
    Raise exception.
    Inherited from abctools.AbjadObject
IntervalObject.__ne__(arg)
    True when id(self) does not equal id(arg).
    Return boolean.
    Inherited from abctools.AbjadObject
IntervalObject.__repr__()
IntervalObject.__str__()
```

pitchtools.IntervalObjectClass



```
class pitchtools.IntervalObjectClass
    New in version 2.0. Interval-class base class.
```

Read-only Properties

```
IntervalObjectClass.number
IntervalObjectClass.storage_format
    Storage format of Abjad object.
    Return string.
    Inherited from abctools.AbjadObject
```

Special Methods

```
IntervalObjectClass.__abs__()
```

`IntervalObjectClass.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`IntervalObjectClass.__float__()`

`IntervalObjectClass.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`IntervalObjectClass.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`IntervalObjectClass.__hash__()`

`IntervalObjectClass.__int__()`

`IntervalObjectClass.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

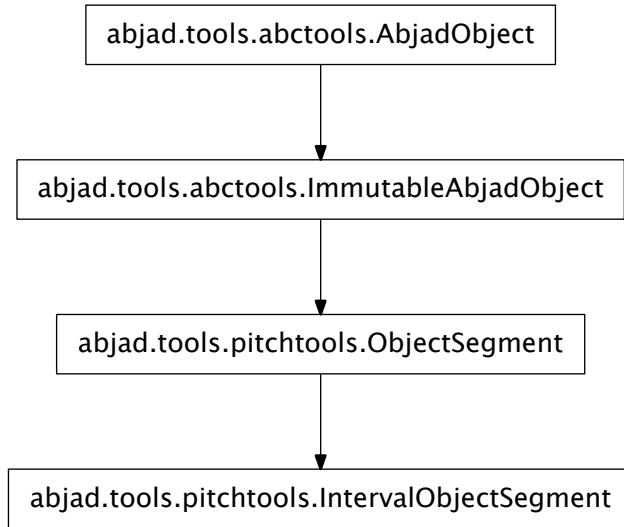
`IntervalObjectClass.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`IntervalObjectClass.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`IntervalObjectClass.__repr__()`

`IntervalObjectClass.__str__()`

pitchtools.IntervalObjectSegment



class `pitchtools.IntervalObjectSegment` (*args, **kwargs)

New in version 2.0. Class of abstract ordered collection of interval instances from which concrete classes inherit.

Read-only Properties

`IntervalObjectSegment.interval_classes`

`IntervalObjectSegment.intervals`

`IntervalObjectSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`IntervalObjectSegment.count` (value) → integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

`IntervalObjectSegment.index` (value[, start[, stop]]) → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`IntervalObjectSegment.rotate` (n)

Special Methods

`IntervalObjectSegment.__add__` (arg)

Inherited from `pitchtools.ObjectSegment`

```
IntervalObjectSegment.__contains__()
    x.__contains__(y) <==> y in x
    Inherited from __builtin__.tuple

IntervalObjectSegment.__eq__()
    x.__eq__(y) <==> x==y
    Inherited from __builtin__.tuple

IntervalObjectSegment.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.tuple

IntervalObjectSegment.__getitem__()
    x.__getitem__(y) <==> x[y]
    Inherited from __builtin__.tuple

IntervalObjectSegment.__getslice__(start, stop)
    Inherited from pitchtools.ObjectSegment

IntervalObjectSegment.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.tuple

IntervalObjectSegment.__hash__() <==> hash(x)
    Inherited from __builtin__.tuple

IntervalObjectSegment.__iter__() <==> iter(x)
    Inherited from __builtin__.tuple

IntervalObjectSegment.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.tuple

IntervalObjectSegment.__len__() <==> len(x)
    Inherited from __builtin__.tuple

IntervalObjectSegment.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple

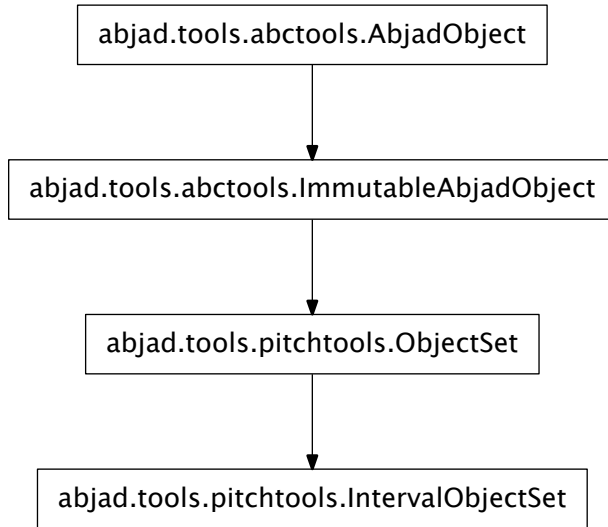
IntervalObjectSegment.__mul__(n)
    Inherited from pitchtools.ObjectSegment

IntervalObjectSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple

IntervalObjectSegment.__repr__()

IntervalObjectSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment

IntervalObjectSegment.__str__()
```


pitchtools.IntervalObjectSet

class `pitchtools.IntervalObjectSet` (*args, **kwargs)
 New in version 2.0. Abstract interval set.

Read-only Properties

`IntervalObjectSet.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`IntervalObjectSet.copy()`

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`

`IntervalObjectSet.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset`

`IntervalObjectSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset`

`IntervalObjectSet.isdisjoint()`
 Return True if two sets have a null intersection.
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.issubset()`
 Report whether another set contains this set.
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.issuperset()`
 Report whether this set contains another set.
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.symmetric_difference()`
 Return the symmetric difference of two sets as a new set.
 (i.e. all elements that are in exactly one of the sets.)
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.union()`
 Return the union of sets as a new set.
 (i.e. all elements that are in either set.)
 Inherited from `__builtin__.frozenset`

Special Methods

`IntervalObjectSet.__and__()`
 $x._and_(y) \iff x \& y$
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.__cmp__()` $\iff cmp(x, y)$
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.__contains__()`
 $x._contains_(y) \iff y \text{ in } x$.
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.__eq__()`
 $x._eq_(y) \iff x == y$
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.__ge__()`
 $x._ge_(y) \iff x \geq y$
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.__gt__()`
 $x._gt_(y) \iff x > y$
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.__hash__()` $\iff hash(x)$
 Inherited from `__builtin__.frozenset`

`IntervalObjectSet.__iter__()` $\iff iter(x)$
 Inherited from `__builtin__.frozenset`

```

IntervalObjectSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset

IntervalObjectSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset

IntervalObjectSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

IntervalObjectSet.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.frozenset

IntervalObjectSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

IntervalObjectSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

IntervalObjectSet.__repr__() <==> repr(x)
    Inherited from __builtin__.frozenset

IntervalObjectSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

IntervalObjectSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

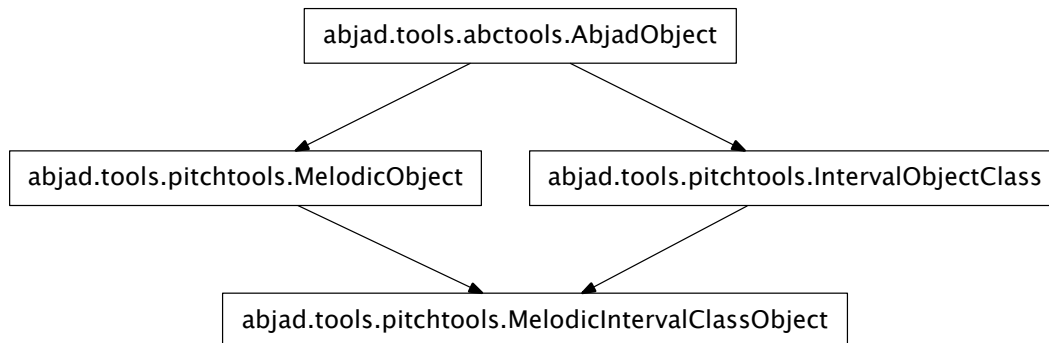
IntervalObjectSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

IntervalObjectSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

IntervalObjectSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset

```

pitchtools.MelodicIntervalClassObject



class `pitchtools.MelodicIntervalClassObject`
 New in version 2.0. Melodic interval-class base class.

Read-only Properties

`MelodicIntervalClassObject.direction_number`

`MelodicIntervalClassObject.direction_symbol`

`MelodicIntervalClassObject.direction_word`

`MelodicIntervalClassObject.number`

Inherited from `pitchtools.IntervalObjectClass`

`MelodicIntervalClassObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`MelodicIntervalClassObject.__abs__()`

Inherited from `pitchtools.IntervalObjectClass`

`MelodicIntervalClassObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MelodicIntervalClassObject.__float__()`

Inherited from `pitchtools.IntervalObjectClass`

`MelodicIntervalClassObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicIntervalClassObject.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`MelodicIntervalClassObject.__hash__()`
 Inherited from `pitchtools.IntervalObjectClass`

`MelodicIntervalClassObject.__int__()`
 Inherited from `pitchtools.IntervalObjectClass`

`MelodicIntervalClassObject.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

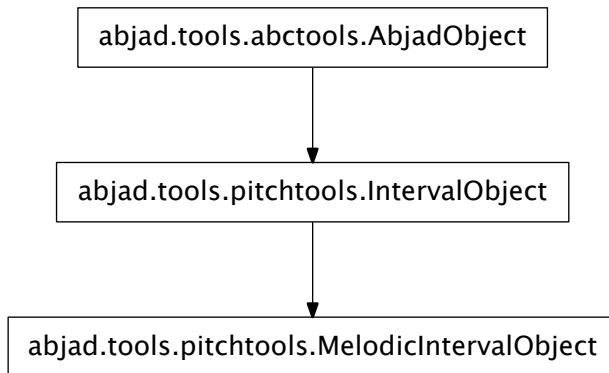
`MelodicIntervalClassObject.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`MelodicIntervalClassObject.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`MelodicIntervalClassObject.__repr__()`
 Inherited from `pitchtools.IntervalObjectClass`

`MelodicIntervalClassObject.__str__()`
 Inherited from `pitchtools.IntervalObjectClass`

pitchtools.MelodicIntervalObject



`class pitchtools.MelodicIntervalObject`
New in version 2.0. Melodic interval base class.

Read-only Properties

`MelodicIntervalObject.cents`
Inherited from `pitchtools.IntervalObject`

`MelodicIntervalObject.direction_number`

`MelodicIntervalObject.direction_string`

`MelodicIntervalObject.interval_class`
Inherited from `pitchtools.IntervalObject`

`MelodicIntervalObject.number`
Inherited from `pitchtools.IntervalObject`

`MelodicIntervalObject.semitones`
Inherited from `pitchtools.IntervalObject`

`MelodicIntervalObject.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`MelodicIntervalObject.__abs__()`

`MelodicIntervalObject.__eq__(arg)`

`MelodicIntervalObject.__float__()`
Inherited from `pitchtools.IntervalObject`

`MelodicIntervalObject.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicIntervalObject.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MelodicIntervalObject.__hash__()`
Inherited from `pitchtools.IntervalObject`

`MelodicIntervalObject.__int__()`
Inherited from `pitchtools.IntervalObject`

`MelodicIntervalObject.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicIntervalObject.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicIntervalObject.__ne__(arg)`

`MelodicIntervalObject.__neg__()`

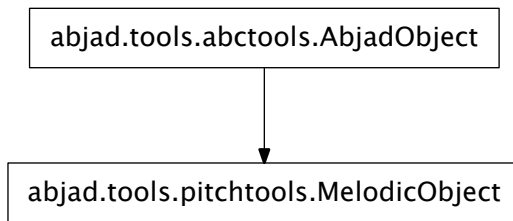
`MelodicIntervalObject.__repr__()`

Inherited from `pitchtools.IntervalObject`

`MelodicIntervalObject.__str__()`

Inherited from `pitchtools.IntervalObject`

`pitchtools.MelodicObject`



class `pitchtools.MelodicObject`

`..versionadded:: 2.0`

Melodic object base class.

Read-only Properties

`MelodicObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`MelodicObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MelodicObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MelodicObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

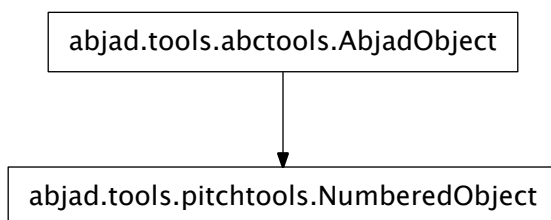
`MelodicObject.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`pitchtools.NumberedObject`



```
class pitchtools.NumberedObject
```

```
..versionadded: 1.1.2
```

Numbered object base class.

Read-only Properties

`NumberedObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`NumberedObject.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`NumberedObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`NumberedObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

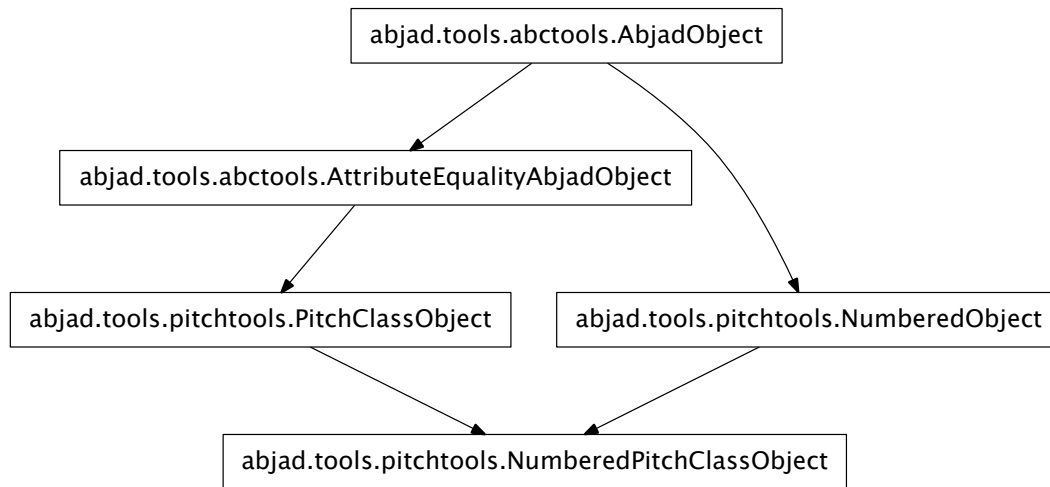
`NumberedObject.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

pitchtools.NumberedPitchClassObject



class `pitchtools.NumberedPitchClassObject`
 New in version 2.0. Numbered pitch-class base class.

Read-only Properties

`NumberedPitchClassObject.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`NumberedPitchClassObject.__eq__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedPitchClassObject.__ge__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedPitchClassObject.__gt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`NumberedPitchClassObject.__hash__()`
 Inherited from `pitchtools.PitchClassObject`

`NumberedPitchClassObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedPitchClassObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedPitchClassObject.__ne__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

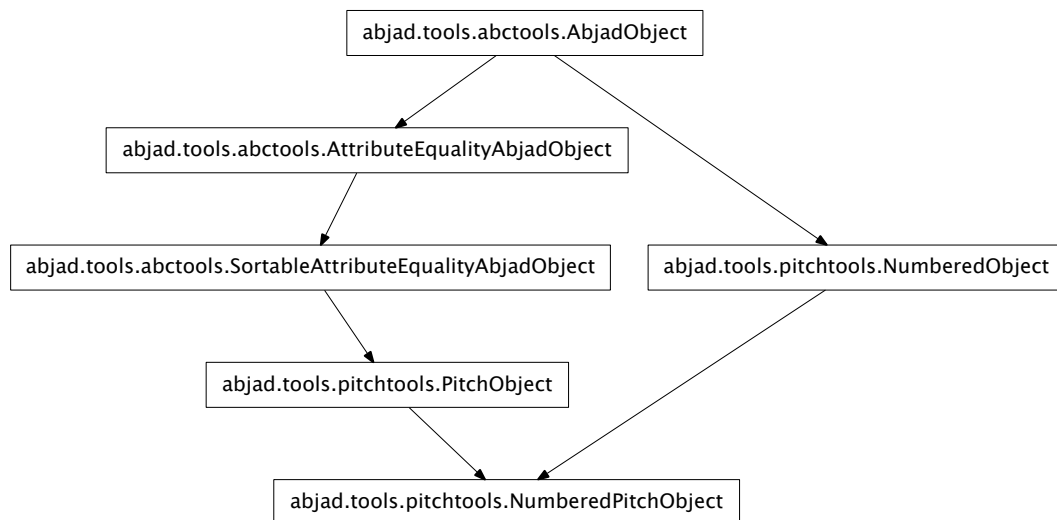
`NumberedPitchClassObject.__repr__()`

Interpreter representation of object defined equal to class name and format string.

Return string.

Inherited from `abctools.AttributeEqualityAbjadObject`

`pitchtools.NumberedPitchObject`



class `pitchtools.NumberedPitchObject`

New in version 2.0. Numbered pitch base class from which concrete classes inherit.

Read-only Properties

`NumberedPitchObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`NumberedPitchObject.__abs__()`

Inherited from `pitchtools.PitchObject`

`NumberedPitchObject.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedPitchObject.__float__()`

Inherited from `pitchtools.PitchObject`

`NumberedPitchObject.__ge__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedPitchObject.__gt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedPitchObject.__hash__()`

Inherited from `pitchtools.PitchObject`

`NumberedPitchObject.__int__()`

Inherited from `pitchtools.PitchObject`

`NumberedPitchObject.__le__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedPitchObject.__lt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

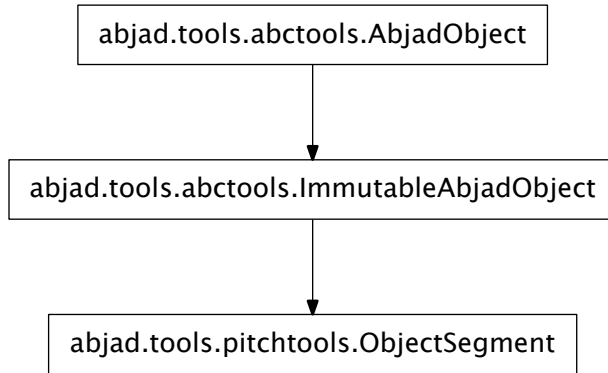
`NumberedPitchObject.__ne__(arg)`

Inherited from `pitchtools.PitchObject`

`NumberedPitchObject.__repr__()`

Inherited from `pitchtools.PitchObject`

pitchtools.ObjectSegment



class `pitchtools.ObjectSegment` (*args, **kwargs)
 New in version 2.0. Mix-in base class for ordered collections of pitch objects.

Read-only Properties

`ObjectSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`ObjectSegment.count` (value) → integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

`ObjectSegment.index` (value[, start[, stop]]) → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

Special Methods

`ObjectSegment.__add__` (arg)

`ObjectSegment.__contains__` ()

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`ObjectSegment.__eq__` ()

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

`ObjectSegment.__ge__` ()

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

```

ObjectSegment.__getitem__ ()
    x.__getitem__(y) <==> x[y]

    Inherited from __builtin__.tuple
ObjectSegment.__getslice__ (start, stop)
ObjectSegment.__gt__ ()
    x.__gt__(y) <==> x>y

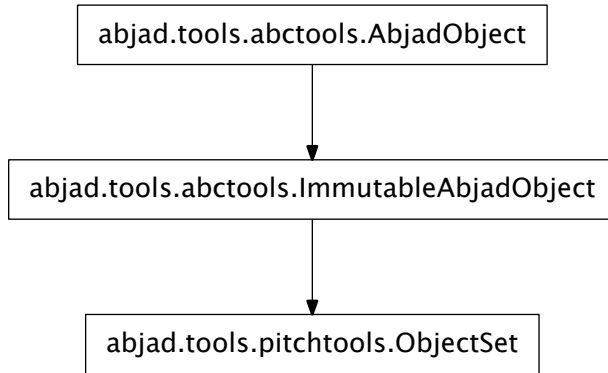
    Inherited from __builtin__.tuple
ObjectSegment.__hash__ () <==> hash(x)
    Inherited from __builtin__.tuple
ObjectSegment.__iter__ () <==> iter(x)
    Inherited from __builtin__.tuple
ObjectSegment.__le__ ()
    x.__le__(y) <==> x<=y

    Inherited from __builtin__.tuple
ObjectSegment.__len__ () <==> len(x)
    Inherited from __builtin__.tuple
ObjectSegment.__lt__ ()
    x.__lt__(y) <==> x<y

    Inherited from __builtin__.tuple
ObjectSegment.__mul__ (n)
ObjectSegment.__ne__ ()
    x.__ne__(y) <==> x!=y

    Inherited from __builtin__.tuple
ObjectSegment.__repr__ () <==> repr(x)
    Inherited from __builtin__.tuple
ObjectSegment.__rmul__ (n)

```

pitchtools.ObjectSet

class `pitchtools.ObjectSet` (*args, **kwargs)
 New in version 2.0. Music-theoretic set base class.

Read-only Properties

`ObjectSet.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`ObjectSet.copy()`
 Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`

`ObjectSet.difference()`
 Return the difference of two or more sets as a new set.
 (i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset`

`ObjectSet.intersection()`
 Return the intersection of two or more sets as a new set.
 (i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset`

`ObjectSet.isdisjoint()`
 Return True if two sets have a null intersection.

Inherited from `__builtin__.frozenset`

`ObjectSet.issubset()`
 Report whether another set contains this set.

Inherited from `__builtin__.frozenset`

`ObjectSet.issuperset()`

Report whether this set contains another set.

Inherited from `__builtin__.frozenset`

`ObjectSet.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Inherited from `__builtin__.frozenset`

`ObjectSet.union()`

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Inherited from `__builtin__.frozenset`

Special Methods

`ObjectSet.__and__()`

`x.__and__(y) <==> x&y`

Inherited from `__builtin__.frozenset`

`ObjectSet.__cmp__(y) <==> cmp(x, y)`

Inherited from `__builtin__.frozenset`

`ObjectSet.__contains__()`

`x.__contains__(y) <==> y in x.`

Inherited from `__builtin__.frozenset`

`ObjectSet.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.frozenset`

`ObjectSet.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.frozenset`

`ObjectSet.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.frozenset`

`ObjectSet.__hash__() <==> hash(x)`

Inherited from `__builtin__.frozenset`

`ObjectSet.__iter__() <==> iter(x)`

Inherited from `__builtin__.frozenset`

`ObjectSet.__le__()`

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.frozenset`

`ObjectSet.__len__() <==> len(x)`

Inherited from `__builtin__.frozenset`


```
ObjectSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

ObjectSet.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.frozenset

ObjectSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

ObjectSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

ObjectSet.__repr__() <==> repr(x)
    Inherited from __builtin__.frozenset

ObjectSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

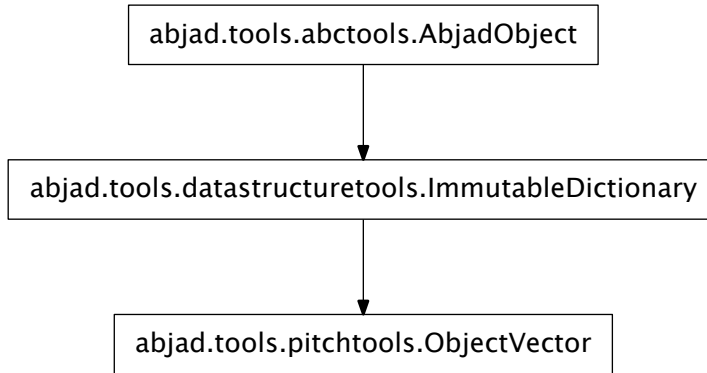
ObjectSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

ObjectSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

ObjectSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

ObjectSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset
```

pitchtools.ObjectVector



class `pitchtools.ObjectVector`

New in version 2.0. Music theoretic vector base class.

Read-only Properties

`ObjectVector.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`ObjectVector.clear()` → None. Remove all items from D.

Inherited from `__builtin__.dict`

`ObjectVector.copy()` → a shallow copy of D

Inherited from `__builtin__.dict`

`ObjectVector.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Inherited from `__builtin__.dict`

`ObjectVector.has_key(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`ObjectVector.items()` → list of D's (key, value) pairs, as 2-tuples

Inherited from `__builtin__.dict`

`ObjectVector.iteritems()` → an iterator over the (key, value) items of D

Inherited from `__builtin__.dict`

`ObjectVector.iterkeys()` → an iterator over the keys of D

Inherited from `__builtin__.dict`

`ObjectVector.itervalues()` → an iterator over the values of D

Inherited from `__builtin__.dict`

`ObjectVector.keys()` → list of D's keys

Inherited from `__builtin__.dict`

`ObjectVector.pop(k[, d])` → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

Inherited from `__builtin__.dict`

`ObjectVector.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Inherited from `__builtin__.dict`

`ObjectVector.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Inherited from `__builtin__.dict`

`ObjectVector.update([E], **F)` → None. Update D from dict/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Inherited from `__builtin__.dict`

`ObjectVector.values()` → list of D's values

Inherited from `__builtin__.dict`

`ObjectVector.viewitems()` → a set-like object providing a view on D's items

Inherited from `__builtin__.dict`

`ObjectVector.viewkeys()` → a set-like object providing a view on D's keys

Inherited from `__builtin__.dict`

`ObjectVector.viewvalues()` → an object providing a view on D's values

Inherited from `__builtin__.dict`

Special Methods

`ObjectVector.__cmp__(y)` <==> `cmp(x, y)`

Inherited from `__builtin__.dict`

`ObjectVector.__contains__(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`ObjectVector.__delitem__(*args)`

Inherited from `datastructuretools.ImmutableDictionary`

`ObjectVector.__eq__()`

`x.__eq__(y)` <==> `x==y`

Inherited from `__builtin__.dict`

`ObjectVector.__ge__()`

`x.__ge__(y)` <==> `x>=y`

Inherited from `__builtin__.dict`

`ObjectVector.__getitem__()`

`x.__getitem__(y)` <==> `x[y]`

Inherited from `__builtin__.dict`

`ObjectVector.__gt__()`

`x.__gt__(y)` <==> `x>y`

Inherited from `__builtin__.dict`

ObjectVector.**__iter__**() $\leq ==>$ *iter(x)*
 Inherited from `__builtin__.dict`

ObjectVector.**__le__**()
 $x.__le__(y) \leq ==> x \leq y$
 Inherited from `__builtin__.dict`

ObjectVector.**__len__**() $\leq ==>$ *len(x)*
 Inherited from `__builtin__.dict`

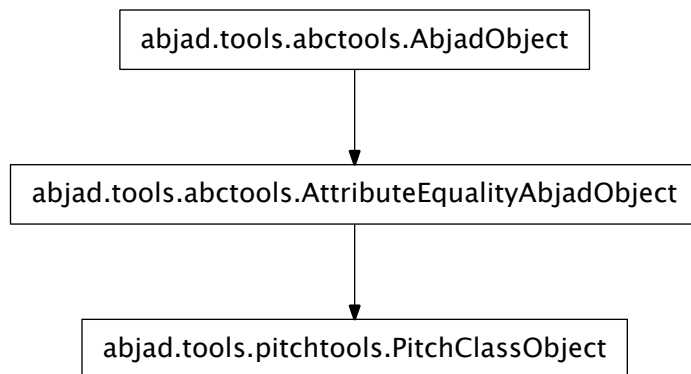
ObjectVector.**__lt__**()
 $x.__lt__(y) \leq ==> x < y$
 Inherited from `__builtin__.dict`

ObjectVector.**__ne__**()
 $x.__ne__(y) \leq ==> x != y$
 Inherited from `__builtin__.dict`

ObjectVector.**__repr__**() $\leq ==>$ *repr(x)*
 Inherited from `__builtin__.dict`

ObjectVector.**__setitem__**(*args)
 Inherited from `datastructuretools.ImmutableDictionary`

pitchtools.PitchClassObject



class `pitchtools.PitchClassObject`
 New in version 2.0. Pitch-class base class.

Read-only Properties

`PitchClassObject.storage_format`
 Storage format of Abjad object.
 Return string.
 Inherited from `abctools.AbjadObject`

Special Methods

`PitchClassObject.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`PitchClassObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PitchClassObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`PitchClassObject.__hash__()`

`PitchClassObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PitchClassObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PitchClassObject.__ne__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

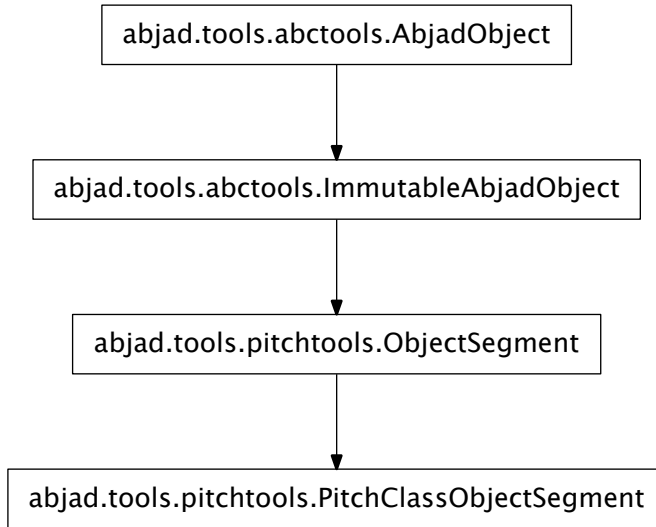
`PitchClassObject.__repr__()`

Interpreter representation of object defined equal to class name and format string.

Return string.

Inherited from `abctools.AttributeEqualityAbjadObject`

`pitchtools.PitchClassObjectSegment`



class `pitchtools.PitchClassObjectSegment` (*args, **kwargs)
 New in version 2.0. Pitch-class segment base class.

Read-only Properties

`PitchClassObjectSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`PitchClassObjectSegment.count` (value) → integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

`PitchClassObjectSegment.index` (value[, start[, stop]]) → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

Special Methods

`PitchClassObjectSegment.__add__` (arg)

Inherited from `pitchtools.ObjectSegment`

`PitchClassObjectSegment.__contains__` ()

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

```

PitchClassObjectSegment.__eq__()
    x.__eq__(y) <==> x==y
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__getitem__()
    x.__getitem__(y) <==> x[y]
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__getslice__(start, stop)
    Inherited from pitchtools.ObjectSegment

PitchClassObjectSegment.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__hash__() <==> hash(x)
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__iter__() <==> iter(x)
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__len__() <==> len(x)
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__mul__(n)
    Inherited from pitchtools.ObjectSegment

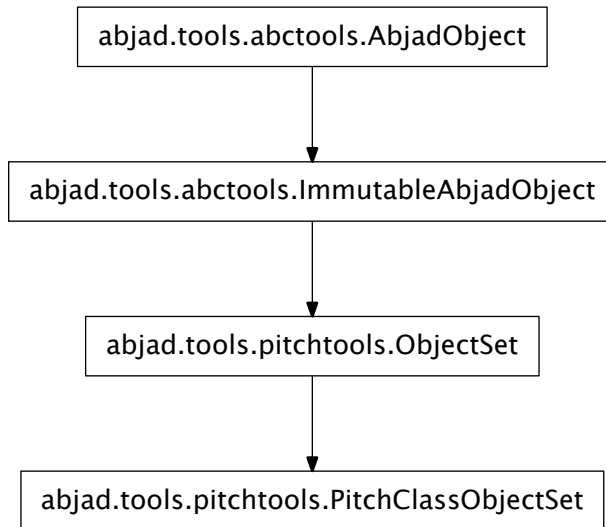
PitchClassObjectSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__repr__() <==> repr(x)
    Inherited from __builtin__.tuple

PitchClassObjectSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment

```

`pitchtools.PitchClassObjectSet`



class `pitchtools.PitchClassObjectSet` (**args, **kwargs*)
 New in version 2.0. Pitch-class set base class.

Read-only Properties

`PitchClassObjectSet.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`PitchClassObjectSet.copy()`

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.isdisjoint()`
 Return True if two sets have a null intersection.
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.issubset()`
 Report whether another set contains this set.
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.issuperset()`
 Report whether this set contains another set.
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.symmetric_difference()`
 Return the symmetric difference of two sets as a new set.
 (i.e. all elements that are in exactly one of the sets.)
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.union()`
 Return the union of sets as a new set.
 (i.e. all elements that are in either set.)
 Inherited from `__builtin__.frozenset`

Special Methods

`PitchClassObjectSet.__and__()`
 $x._and_(y) \iff x \& y$
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.__cmp__()` $\iff cmp(x, y)$
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.__contains__()`
 $x._contains_(y) \iff y \text{ in } x$.
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.__eq__()`
 $x._eq_(y) \iff x == y$
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.__ge__()`
 $x._ge_(y) \iff x \geq y$
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.__gt__()`
 $x._gt_(y) \iff x > y$
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.__hash__()` $\iff hash(x)$
 Inherited from `__builtin__.frozenset`

`PitchClassObjectSet.__iter__()` $\iff iter(x)$
 Inherited from `__builtin__.frozenset`

```

PitchClassObjectSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__repr__() <==> repr(x)
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

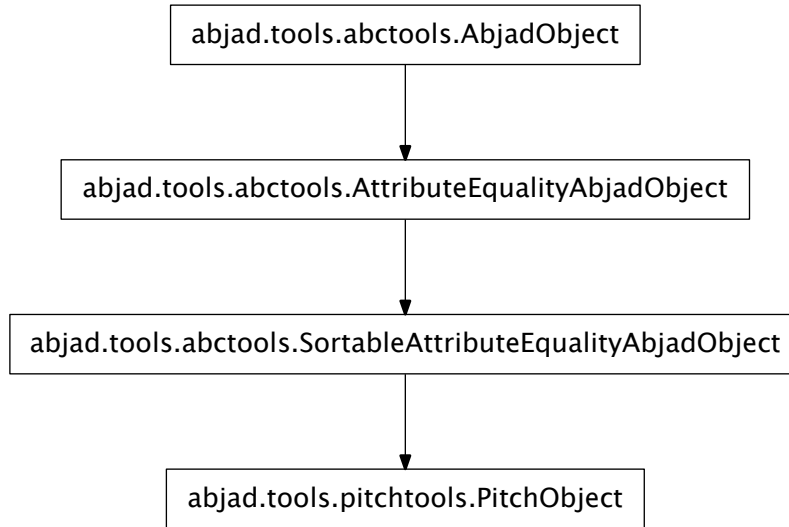
PitchClassObjectSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

PitchClassObjectSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset

```

pitchtools.PitchObject



class `pitchtools.PitchObject`
 New in version 2.0. Pitch base class.

Read-only Properties

`PitchObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`PitchObject.__abs__()`

`PitchObject.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`PitchObject.__float__()`

`PitchObject.__ge__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`PitchObject.__gt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`PitchObject.__hash__()`

`PitchObject.__int__()`

`PitchObject.__le__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`PitchObject.__lt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

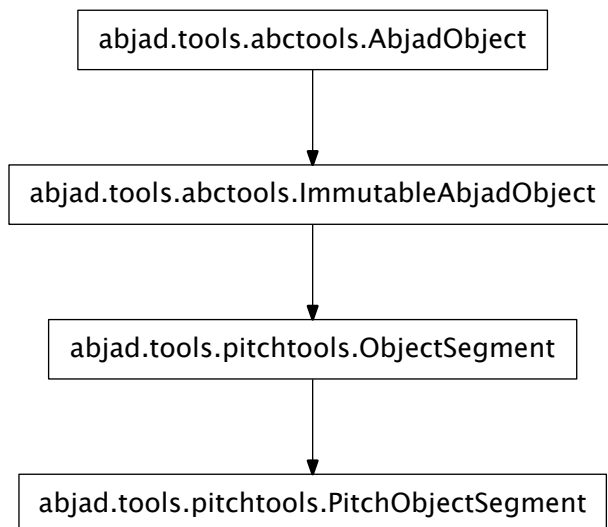
Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`PitchObject.__ne__(arg)`

`PitchObject.__repr__()`

`pitchtools.PitchObjectSegment`



`class pitchtools.PitchObjectSegment(*args, **kwargs)`

New in version 2.0. Pitch segment base class.

Read-only Properties

PitchObjectSegment.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

PitchObjectSegment.**count**(*value*) → integer – return number of occurrences of *value*

Inherited from `__builtin__.tuple`

PitchObjectSegment.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of *value*.

Raises `ValueError` if the *value* is not present.

Inherited from `__builtin__.tuple`

Special Methods

PitchObjectSegment.**__add__**(*arg*)

Inherited from `pitchtools.ObjectSegment`

PitchObjectSegment.**__contains__**()

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

PitchObjectSegment.**__eq__**()

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

PitchObjectSegment.**__ge__**()

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

PitchObjectSegment.**__getitem__**()

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

PitchObjectSegment.**__getslice__**(*start*, *stop*)

Inherited from `pitchtools.ObjectSegment`

PitchObjectSegment.**__gt__**()

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.tuple`

PitchObjectSegment.**__hash__**() <==> *hash(x)*

Inherited from `__builtin__.tuple`

PitchObjectSegment.**__iter__**() <==> *iter(x)*

Inherited from `__builtin__.tuple`

PitchObjectSegment.**__le__**()

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.tuple`

PitchObjectSegment.**__len__**() <==> *len(x)*

Inherited from `__builtin__.tuple`

```
PitchObjectSegment.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple

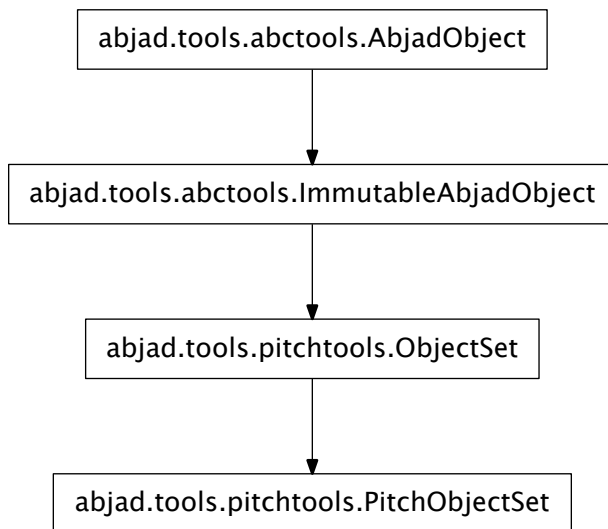
PitchObjectSegment.__mul__(n)
    Inherited from pitchtools.ObjectSegment

PitchObjectSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple

PitchObjectSegment.__repr__() <==> repr(x)
    Inherited from __builtin__.tuple

PitchObjectSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment
```

pitchtools.PitchObjectSet



```
class pitchtools.PitchObjectSet (*args, **kwargs)
    New in version 2.0. Pitch set base class.
```

Read-only Properties

```
PitchObjectSet.storage_format
    Storage format of Abjad object.
    Return string.
    Inherited from abctools.AbjadObject
```

Methods`PitchObjectSet.copy()`

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset``PitchObjectSet.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset``PitchObjectSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset``PitchObjectSet.isdisjoint()`

Return True if two sets have a null intersection.

Inherited from `__builtin__.frozenset``PitchObjectSet.issubset()`

Report whether another set contains this set.

Inherited from `__builtin__.frozenset``PitchObjectSet.issuperset()`

Report whether this set contains another set.

Inherited from `__builtin__.frozenset``PitchObjectSet.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Inherited from `__builtin__.frozenset``PitchObjectSet.union()`

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Inherited from `__builtin__.frozenset`**Special Methods**`PitchObjectSet.__and__()``x.__and__(y) <==> x&y`Inherited from `__builtin__.frozenset``PitchObjectSet.__cmp__(y) <==> cmp(x, y)`Inherited from `__builtin__.frozenset``PitchObjectSet.__contains__()``x.__contains__(y) <==> y in x.`Inherited from `__builtin__.frozenset``PitchObjectSet.__eq__()``x.__eq__(y) <==> x==y`Inherited from `__builtin__.frozenset`

```

PitchObjectSet.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.frozenset

PitchObjectSet.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.frozenset

PitchObjectSet.__hash__() <==> hash(x)
    Inherited from __builtin__.frozenset

PitchObjectSet.__iter__() <==> iter(x)
    Inherited from __builtin__.frozenset

PitchObjectSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset

PitchObjectSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset

PitchObjectSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

PitchObjectSet.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.frozenset

PitchObjectSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

PitchObjectSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

PitchObjectSet.__repr__() <==> repr(x)
    Inherited from __builtin__.frozenset

PitchObjectSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

PitchObjectSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

PitchObjectSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

PitchObjectSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

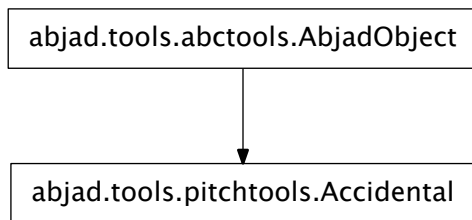
```



```
PitchObjectSet.__xor__ ()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset
```

concrete classes

pitchtools.Accidental



```
class pitchtools.Accidental (arg='')
    New in version 2.0. Abjad model of the accidental:
```

```
>>> pitchtools.Accidental('s')
Accidental('s')
```

Accidentals are immutable.

Read-only Properties

Accidental.alphabetic_accidental_abbreviation

Read-only alphabetic string:

```
>>> accidental = pitchtools.Accidental('s')
>>> accidental.alphabetic_accidental_abbreviation
's'
```

Return string.

Accidental.is_adjusted

True for all accidentals equal to a nonzero number of semitones. False otherwise:

```
>>> accidental = pitchtools.Accidental('s')
>>> accidental.is_adjusted
True
```

Return boolean.

Accidental.lilypond_format

Read-only LilyPond input format of accidental:

```
>>> accidental = pitchtools.Accidental('s')
>>> accidental.lilypond_format
's'
```

Return string.

Accidental.name

Read-only name of accidental:

```
>>> accidental = pitchtools.Accidental('s')
>>> accidental.name
'sharp'
```

Return string.

Accidental.semitones

Read-only semitones of accidental:

```
>>> accidental = pitchtools.Accidental('s')
>>> accidental.semitones
1
```

Return number.

Accidental.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Accidental.symbolic_accidental_string

Read-only symbolic string of accidental:

```
>>> accidental = pitchtools.Accidental('s')
>>> accidental.symbolic_accidental_string
'#'
```

Return string.

Special Methods

`Accidental.__add__(arg)`

`Accidental.__eq__(arg)`

`Accidental.__ge__(arg)`

`Accidental.__gt__(arg)`

`Accidental.__le__(arg)`

`Accidental.__lt__(arg)`

`Accidental.__ne__(arg)`

`Accidental.__neg__()`

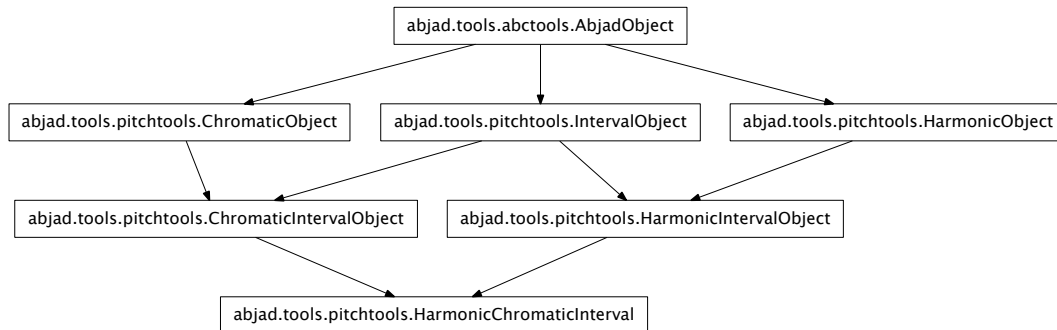
`Accidental.__nonzero__()`

`Accidental.__repr__()`

`Accidental.__str__()`

`Accidental.__sub__(arg)`

pitchtools.HarmonicChromaticInterval



class `pitchtools.HarmonicChromaticInterval` (*arg*)
 New in version 2.0. Abjad model of harmonic chromatic interval:

```
>>> pitchtools.HarmonicChromaticInterval(-14)
HarmonicChromaticInterval(14)
```

Harmonic chromatic intervals are immutable.

Read-only Properties

`HarmonicChromaticInterval.cents`

Inherited from `pitchtools.IntervalObject`

`HarmonicChromaticInterval.harmonic_chromatic_interval_class`

Read-only harmonic chromatic interval-class:

```
>>> harmonic_chromatic_interval = pitchtools.HarmonicChromaticInterval(14)
>>> harmonic_chromatic_interval.harmonic_chromatic_interval_class
HarmonicChromaticIntervalClass(2)
```

Return harmonic chromatic interval-class.

`HarmonicChromaticInterval.interval_class`

Inherited from `pitchtools.IntervalObject`

`HarmonicChromaticInterval.number`

Inherited from `pitchtools.ChromaticIntervalObject`

`HarmonicChromaticInterval.semitones`

Inherited from `pitchtools.ChromaticIntervalObject`

`HarmonicChromaticInterval.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

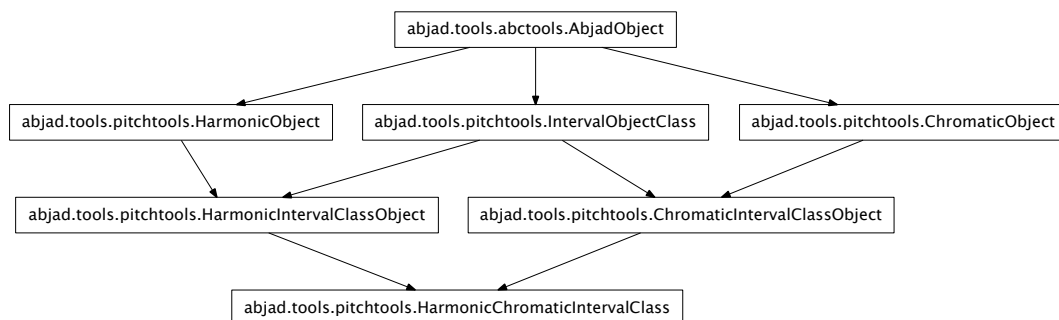
Special Methods

```

HarmonicChromaticInterval.__abs__()
    Inherited from pitchtools.ChromaticIntervalObject
HarmonicChromaticInterval.__add__(arg)
    Inherited from pitchtools.ChromaticIntervalObject
HarmonicChromaticInterval.__eq__(arg)
    Inherited from pitchtools.ChromaticIntervalObject
HarmonicChromaticInterval.__float__()
    Inherited from pitchtools.ChromaticIntervalObject
HarmonicChromaticInterval.__ge__(arg)
HarmonicChromaticInterval.__gt__(arg)
HarmonicChromaticInterval.__hash__()
    Inherited from pitchtools.IntervalObject
HarmonicChromaticInterval.__int__()
    Inherited from pitchtools.ChromaticIntervalObject
HarmonicChromaticInterval.__le__(arg)
HarmonicChromaticInterval.__lt__(arg)
HarmonicChromaticInterval.__ne__(arg)
    Inherited from pitchtools.ChromaticIntervalObject
HarmonicChromaticInterval.__repr__()
    Inherited from pitchtools.ChromaticIntervalObject
HarmonicChromaticInterval.__str__()
    Inherited from pitchtools.ChromaticIntervalObject
HarmonicChromaticInterval.__sub__(arg)
    Inherited from pitchtools.ChromaticIntervalObject

```

pitchtools.HarmonicChromaticIntervalClass



```

class pitchtools.HarmonicChromaticIntervalClass (token)
    New in version 2.0. Abjad model of harmonic chromatic interval-class:

```

```
>>> pitchtools.HarmonicChromaticIntervalClass(-14)
HarmonicChromaticIntervalClass(2)
```

Harmonic chromatic interval-classes are immutable.

Read-only Properties

`HarmonicChromaticIntervalClass.number`

Inherited from `pitchtools.IntervalObjectClass`

`HarmonicChromaticIntervalClass.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`HarmonicChromaticIntervalClass.__abs__()`

Inherited from `pitchtools.ChromaticIntervalClassObject`

`HarmonicChromaticIntervalClass.__eq__(arg)`

`HarmonicChromaticIntervalClass.__float__()`

Inherited from `pitchtools.ChromaticIntervalClassObject`

`HarmonicChromaticIntervalClass.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HarmonicChromaticIntervalClass.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`HarmonicChromaticIntervalClass.__hash__()`

Inherited from `pitchtools.IntervalObjectClass`

`HarmonicChromaticIntervalClass.__int__()`

Inherited from `pitchtools.ChromaticIntervalClassObject`

`HarmonicChromaticIntervalClass.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HarmonicChromaticIntervalClass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

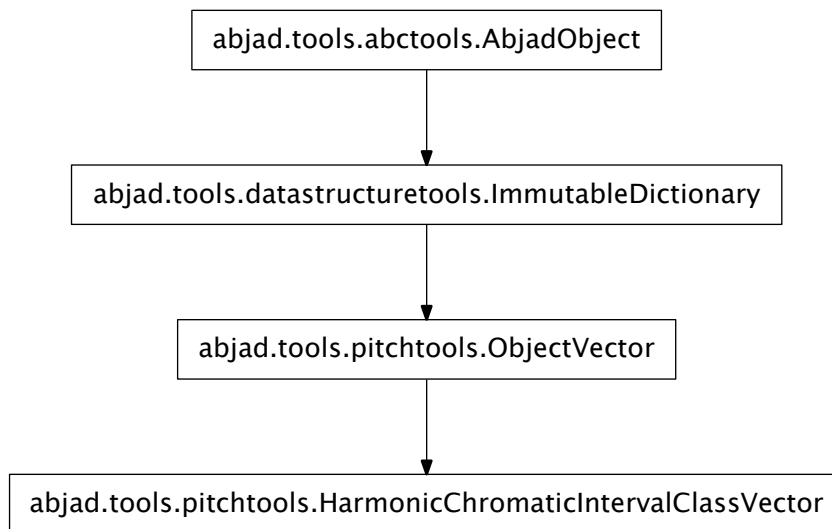
`HarmonicChromaticIntervalClass.__ne__(arg)`

`HarmonicChromaticIntervalClass.__repr__()`

Inherited from `pitchtools.IntervalObjectClass`

HarmonicChromaticIntervalClass.__str__()
 Inherited from pitchtools.IntervalObjectClass

pitchtools.HarmonicChromaticIntervalClassVector



class pitchtools.**HarmonicChromaticIntervalClassVector**(*expr*)
 New in version 2.0. Abjad model of harmonic chromatic interval-class vector:

```

>>> staff = Staff("c'8 d'8 e'8 f'8 g'8")
>>> hcicv = pitchtools.HarmonicChromaticIntervalClassVector(staff)
>>> print hcicv
0 1 3 2 1 2 0 1 0 0 0 0
    
```

Harmonic chromatic interval-class vector is quartertone-aware:

```

>>> staff.append(Note(1.5, (1, 4)))
>>> hcicv = pitchtools.HarmonicChromaticIntervalClassVector(staff)
>>> print hcicv
0 1 3 2 1 2 0 1 0 0 0 0
1 1 1 1 0 1 0 0 0 0 0 0
    
```

Harmonic chromatic interval-class vectors are immutable.

Read-only Properties

HarmonicChromaticIntervalClassVector.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from abctools.AbjadObject

Methods

`HarmonicChromaticIntervalClassVector.clear()` → None. Remove all items from D.

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.copy()` → a shallow copy of D

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.has_key(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.has_none_of(chromatic_interval_numbers)`
True when harmonic chromatic interval-class vector contains none of *chromatic_interval_numbers*. Otherwise false:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8")

>>> hcicv = pitchtools.HarmonicChromaticIntervalClassVector(staff)
>>> hcicv.has_none_of([9, 10, 11])
True
```

Return boolean.

`HarmonicChromaticIntervalClassVector.items()` → list of D's (key, value) pairs, as 2-tuples

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.iteritems()` → an iterator over the (key, value) items of D

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.iterkeys()` → an iterator over the keys of D

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.itervalues()` → an iterator over the values of D

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.keys()` → list of D's keys

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.pop(k[, d])` → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.popitem()` → (k, v), remove and return some (key, value) pair as a

2-tuple; but raise `KeyError` if D is empty.

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.update([E], **F)` → None. Update D from dict/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.values()` → list of D's values

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.viewitems()` → a set-like object providing a view on D's items

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.viewkeys()` → a set-like object providing a view on D's keys

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.viewvalues()` → an object providing a view on D's values

Inherited from `__builtin__.dict`

Special Methods

`HarmonicChromaticIntervalClassVector.__cmp__(y) <==> cmp(x, y)`

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__contains__(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__delitem__(*args)`

Inherited from `datastructuretools.ImmutableDictionary`

`HarmonicChromaticIntervalClassVector.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__iter__()` <==> `iter(x)`

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__le__()`

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__len__()` <==> `len(x)`

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__lt__()`

`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__ne__()`

`x.__ne__(y) <==> x!=y`

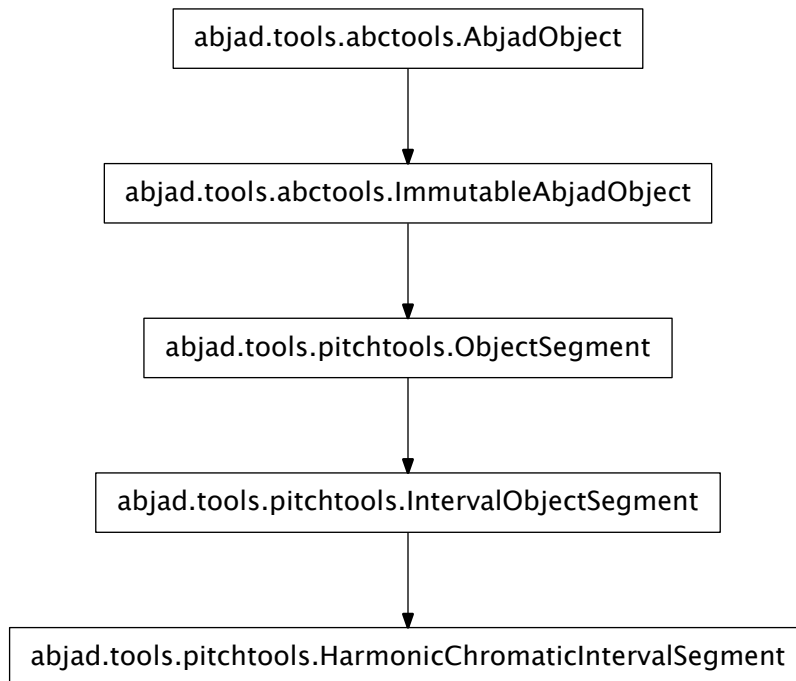
Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__repr__()`

`HarmonicChromaticIntervalClassVector.__setitem__(*args)`

`HarmonicChromaticIntervalClassVector.__str__()`

`pitchtools.HarmonicChromaticIntervalSegment`



class `pitchtools.HarmonicChromaticIntervalSegment` *(*args, **kwargs)*

New in version 2.0. Abjad model of harmonic chromatic interval segment:

```
>>> pitchtools.HarmonicChromaticIntervalSegment([10, -12, -13, -13.5])
HarmonicChromaticIntervalSegment(10, 12, 13, 13.5)
```

Harmonic chromatic interval segments are immutable.

Read-only Properties

`HarmonicChromaticIntervalSegment.interval_classes`

Inherited from `pitchtools.IntervalObjectSegment`

`HarmonicChromaticIntervalSegment.intervals`

Inherited from `pitchtools.IntervalObjectSegment`

`HarmonicChromaticIntervalSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`HarmonicChromaticIntervalSegment.count(value) → integer` – return number of occurrences of value

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.index(value[, start[, stop]]) → integer` – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.rotate(n)`

Inherited from `pitchtools.IntervalObjectSegment`

Special Methods

`HarmonicChromaticIntervalSegment.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`HarmonicChromaticIntervalSegment.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__getslice__(start, stop)`

Inherited from `pitchtools.ObjectSegment`

`HarmonicChromaticIntervalSegment.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__hash__() <==> hash(x)`

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__iter__() <==> iter(x)`

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__le__()`

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__len__() <==> len(x)`

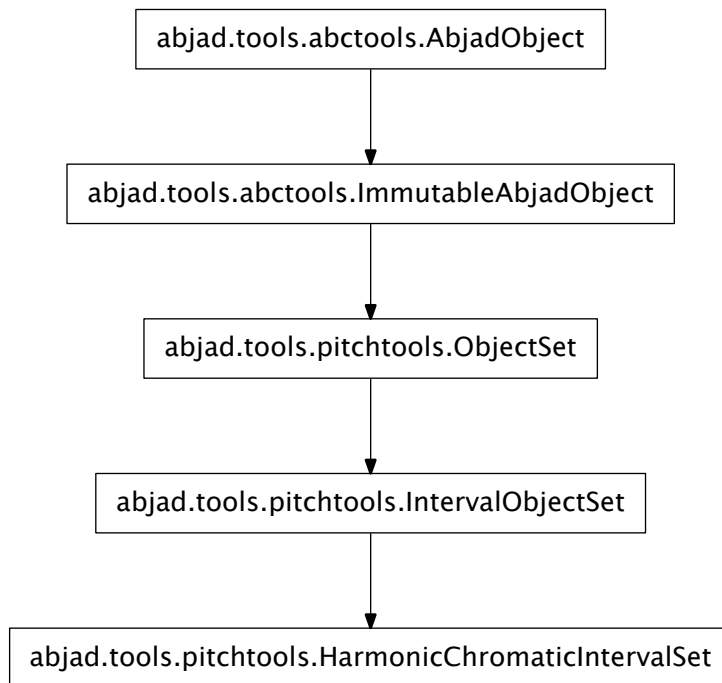
Inherited from `__builtin__.tuple`

```

HarmonicChromaticIntervalSegment.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple
HarmonicChromaticIntervalSegment.__mul__(n)
    Inherited from pitchtools.ObjectSegment
HarmonicChromaticIntervalSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple
HarmonicChromaticIntervalSegment.__repr__()
    Inherited from pitchtools.IntervalObjectSegment
HarmonicChromaticIntervalSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment
HarmonicChromaticIntervalSegment.__str__()
    Inherited from pitchtools.IntervalObjectSegment

```

pitchtools.HarmonicChromaticIntervalSet



```

class pitchtools.HarmonicChromaticIntervalSet (*args, **kwargs)
    New in version 2.0. Abjad model of harmonic chromatic interval set:

```

```
>>> pitchtools.HarmonicChromaticIntervalSet([10, -12, -13, -13, -13.5])
HarmonicChromaticIntervalSet(10, 12, 13, 13.5)
```

Harmonic chromatic interval sets are immutable.

Read-only Properties

HarmonicChromaticIntervalSet.**harmonic_chromatic_interval_numbers**

HarmonicChromaticIntervalSet.**harmonic_chromatic_intervals**

HarmonicChromaticIntervalSet.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

HarmonicChromaticIntervalSet.**copy()**

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**difference()**

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**intersection()**

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**isdisjoint()**

Return True if two sets have a null intersection.

Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**issubset()**

Report whether another set contains this set.

Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**issuperset()**

Report whether this set contains another set.

Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**symmetric_difference()**

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**union()**

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Inherited from `__builtin__.frozenset`

Special Methods

```

HarmonicChromaticIntervalSet.__and__()
    x.__and__(y) <==> x&y

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__cmp__(y) <==> cmp(x, y)
    Inherited from __builtin__.frozenset

HarmonicChromaticIntervalSet.__contains__()
    x.__contains__(y) <==> y in x.

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__eq__()
    x.__eq__(y) <==> x==y

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__ge__()
    x.__ge__(y) <==> x>=y

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__gt__()
    x.__gt__(y) <==> x>y

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__hash__() <==> hash(x)
    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__iter__() <==> iter(x)
    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__le__()
    x.__le__(y) <==> x<=y

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__lt__()
    x.__lt__(y) <==> x<y

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__ne__()
    x.__ne__(y) <==> x!=y

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__or__()
    x.__or__(y) <==> x|y

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__rand__()
    x.__rand__(y) <==> y&x

    Inherited from __builtin__.frozenset
HarmonicChromaticIntervalSet.__repr__()

```

```

HarmonicChromaticIntervalSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

HarmonicChromaticIntervalSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

HarmonicChromaticIntervalSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

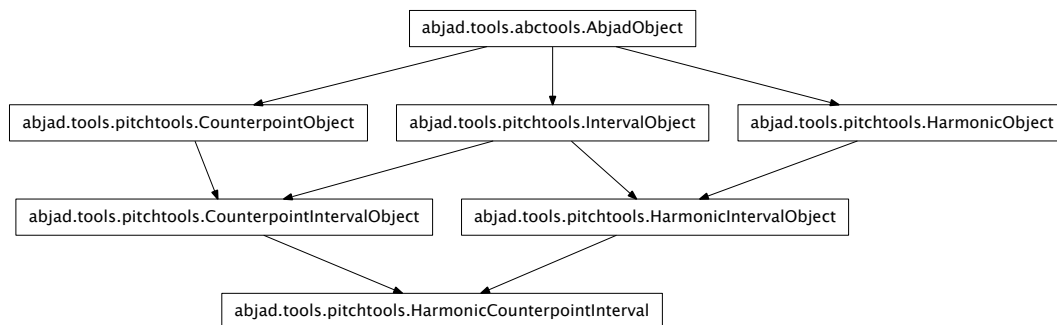
HarmonicChromaticIntervalSet.__str__()

HarmonicChromaticIntervalSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

HarmonicChromaticIntervalSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset

```

pitchtools.HarmonicCounterpointInterval



class `pitchtools.HarmonicCounterpointInterval` (*token*)
 New in version 2.0. Abjad model of harmonic counterpoint interval:

```

>>> pitchtools.HarmonicCounterpointInterval(-9)
HarmonicCounterpointInterval(9)

```

Harmonic counterpoint intervals are immutable.

Read-only Properties

```

HarmonicCounterpointInterval.cents
    Inherited from pitchtools.IntervalObject
HarmonicCounterpointInterval.harmonic_counterpoint_interval_class

```

HarmonicCounterpointInterval.**interval_class**

Inherited from `pitchtools.IntervalObject`

HarmonicCounterpointInterval.**number**

Inherited from `pitchtools.CounterpointIntervalObject`

HarmonicCounterpointInterval.**semitones**

Inherited from `pitchtools.CounterpointIntervalObject`

HarmonicCounterpointInterval.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

HarmonicCounterpointInterval.**__abs__**()

Inherited from `pitchtools.CounterpointIntervalObject`

HarmonicCounterpointInterval.**__eq__**(arg)

HarmonicCounterpointInterval.**__float__**()

Inherited from `pitchtools.CounterpointIntervalObject`

HarmonicCounterpointInterval.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

HarmonicCounterpointInterval.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

HarmonicCounterpointInterval.**__hash__**()

Inherited from `pitchtools.IntervalObject`

HarmonicCounterpointInterval.**__int__**()

Inherited from `pitchtools.CounterpointIntervalObject`

HarmonicCounterpointInterval.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

HarmonicCounterpointInterval.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

HarmonicCounterpointInterval.**__ne__**(arg)

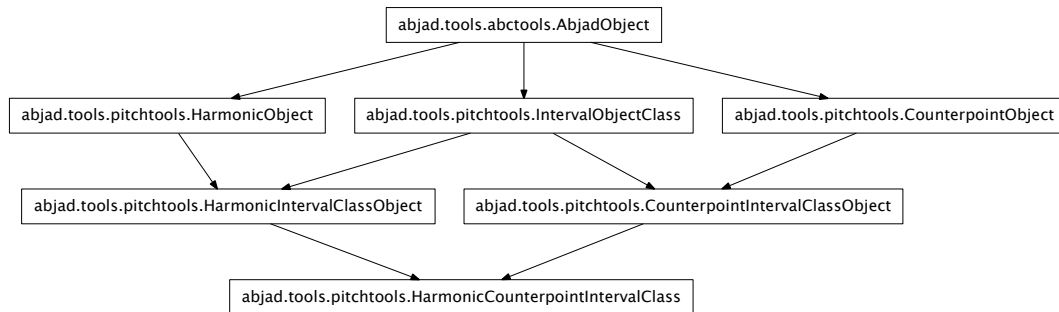
HarmonicCounterpointInterval.**__repr__**()

Inherited from `pitchtools.IntervalObject`

HarmonicCounterpointInterval.**__str__**()

Inherited from `pitchtools.IntervalObject`

pitchtools.HarmonicCounterpointIntervalClass



class `pitchtools.HarmonicCounterpointIntervalClass` (*token*)

New in version 2.0. Abjad model of harmonic counterpoint interval-class:

```
>>> pitchtools.HarmonicCounterpointIntervalClass(-9)
HarmonicCounterpointIntervalClass(2)
```

Harmonic counterpoint interval-classes are immutable.

Read-only Properties

`HarmonicCounterpointIntervalClass.number`

Inherited from `pitchtools.IntervalObjectClass`

`HarmonicCounterpointIntervalClass.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`HarmonicCounterpointIntervalClass.__abs__()`

Inherited from `pitchtools.CounterpointIntervalClassObject`

`HarmonicCounterpointIntervalClass.__eq__(arg)`

`HarmonicCounterpointIntervalClass.__float__()`

Inherited from `pitchtools.CounterpointIntervalClassObject`

`HarmonicCounterpointIntervalClass.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HarmonicCounterpointIntervalClass.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`HarmonicCounterpointIntervalClass.__hash__()`
 Inherited from `pitchtools.IntervalObjectClass`

`HarmonicCounterpointIntervalClass.__int__()`
 Inherited from `pitchtools.CounterpointIntervalClassObject`

`HarmonicCounterpointIntervalClass.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

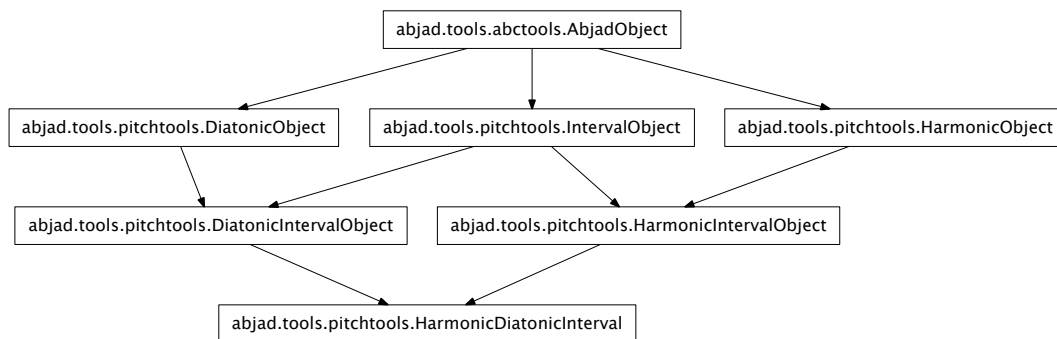
`HarmonicCounterpointIntervalClass.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`HarmonicCounterpointIntervalClass.__ne__(arg)`

`HarmonicCounterpointIntervalClass.__repr__()`
 Inherited from `pitchtools.IntervalObjectClass`

`HarmonicCounterpointIntervalClass.__str__()`
 Inherited from `pitchtools.IntervalObjectClass`

`pitchtools.HarmonicDiatonicInterval`



```

class pitchtools.HarmonicDiatonicInterval(*args)
    New in version 2.0. Abjad model harmonic diatonic interval:

    >>> pitchtools.HarmonicDiatonicInterval('M9')
    HarmonicDiatonicInterval('M9')
    
```

Harmonic diatonic intervals are immutable.

Read-only Properties

`HarmonicDiatonicInterval.cents`
 Inherited from `pitchtools.IntervalObject`

`HarmonicDiatonicInterval.diatonic_interval_class`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.harmonic_counterpoint_interval`

`HarmonicDiatonicInterval.harmonic_diatonic_interval_class`

`HarmonicDiatonicInterval.interval_class`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.interval_string`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.melodic_diatonic_interval_ascending`

`HarmonicDiatonicInterval.melodic_diatonic_interval_descending`

`HarmonicDiatonicInterval.number`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.quality_string`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.semitones`

`HarmonicDiatonicInterval.staff_spaces`

`HarmonicDiatonicInterval.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`HarmonicDiatonicInterval.__abs__()`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.__eq__(arg)`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.__float__()`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.__ge__(arg)`

`HarmonicDiatonicInterval.__gt__(arg)`

`HarmonicDiatonicInterval.__hash__()`
Inherited from `pitchtools.IntervalObject`

`HarmonicDiatonicInterval.__int__()`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.__le__(arg)`

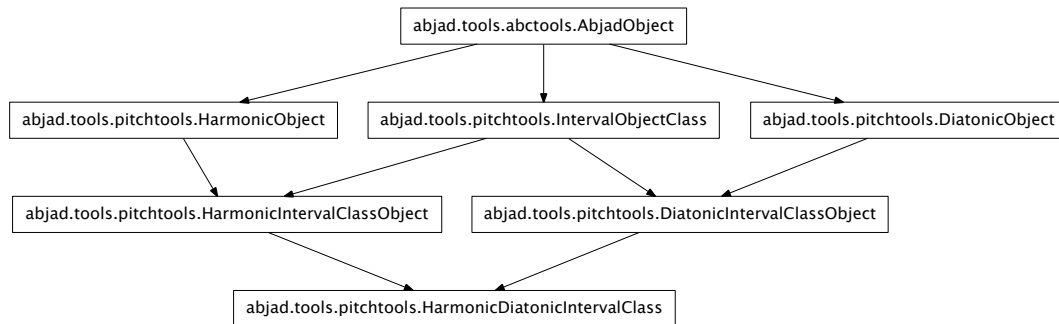
`HarmonicDiatonicInterval.__lt__(arg)`

`HarmonicDiatonicInterval.__ne__(arg)`
Inherited from `pitchtools.DiatonicIntervalObject`

`HarmonicDiatonicInterval.__repr__()`

`HarmonicDiatonicInterval.__str__()`

pitchtools.HarmonicDiatonicIntervalClass



```
class pitchtools.HarmonicDiatonicIntervalClass(*args)
```

New in version 2.0. Abjad model harmonic diatonic interval-class:

```
>>> pitchtools.HarmonicDiatonicIntervalClass('-M9')
HarmonicDiatonicIntervalClass('M2')
```

Harmonic diatonic interval-classes are immutable.

Read-only Properties

HarmonicDiatonicIntervalClass.**number**

Inherited from pitchtools.IntervalObjectClass

HarmonicDiatonicIntervalClass.**quality_string**

Inherited from pitchtools.DiatonicIntervalClassObject

HarmonicDiatonicIntervalClass.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from abctools.AbjadObject

Methods

HarmonicDiatonicIntervalClass.**invert**()

Read-only inversion of harmonic diatonic interval-class:

```
>>> hdic = pitchtools.HarmonicDiatonicIntervalClass('major', -9)
>>> hdic.invert()
HarmonicDiatonicIntervalClass('m7')
```

Return harmonic diatonic interval-class.

Special Methods

HarmonicDiatonicIntervalClass.**__abs__**()

Inherited from pitchtools.DiatonicIntervalClassObject

HarmonicDiatonicIntervalClass.**__eq__**(arg)

HarmonicDiatonicIntervalClass.__float__()
Inherited from pitchtools.DiatonicIntervalClassObject

HarmonicDiatonicIntervalClass.__ge__(arg)
Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

HarmonicDiatonicIntervalClass.__gt__(arg)
Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

HarmonicDiatonicIntervalClass.__hash__()
Inherited from pitchtools.IntervalObjectClass

HarmonicDiatonicIntervalClass.__int__()
Inherited from pitchtools.DiatonicIntervalClassObject

HarmonicDiatonicIntervalClass.__le__(arg)
Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

HarmonicDiatonicIntervalClass.__lt__(arg)
Abjad objects by default do not implement this method.

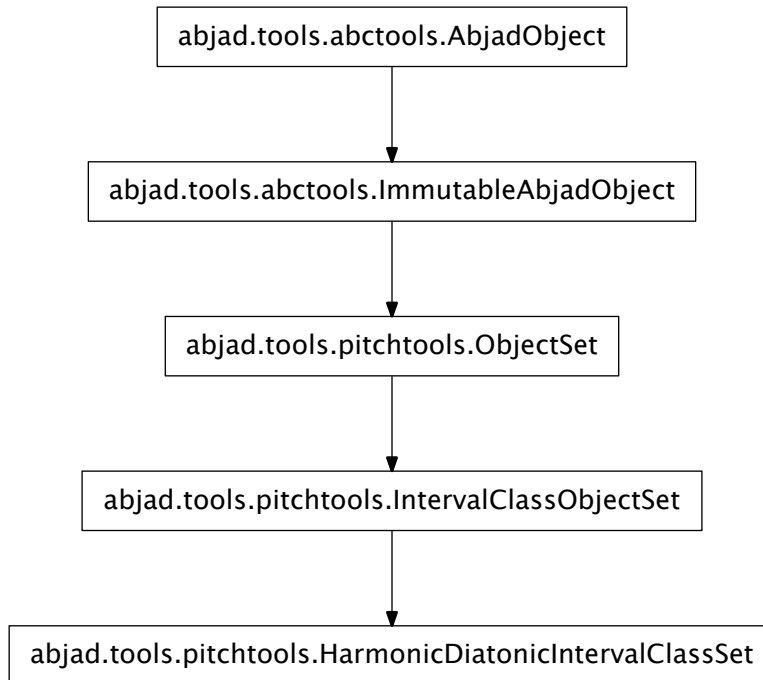
Raise exception.

Inherited from abctools.AbjadObject

HarmonicDiatonicIntervalClass.__ne__(arg)

HarmonicDiatonicIntervalClass.__repr__()
Inherited from pitchtools.DiatonicIntervalClassObject

HarmonicDiatonicIntervalClass.__str__()

pitchtools.HarmonicDiatonicIntervalClassSet

class `pitchtools.HarmonicDiatonicIntervalClassSet` (*args, **kwargs)

New in version 2.0. Abjad model of harmonic diatonic interval-class set:

```
>>> pitchtools.HarmonicDiatonicIntervalClassSet('m2 M2 m3 M3')
HarmonicDiatonicIntervalClassSet('m2 M2 m3 M3')
```

Harmonic diatonic interval-class sets are immutable.

Read-only Properties

`HarmonicDiatonicIntervalClassSet.harmonic_diatonic_interval_classes`

`HarmonicDiatonicIntervalClassSet.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`HarmonicDiatonicIntervalClassSet.copy()`

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.isdisjoint()`

Return True if two sets have a null intersection.

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.issubset()`

Report whether another set contains this set.

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.issuperset()`

Report whether this set contains another set.

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.union()`

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Inherited from `__builtin__.frozenset`

Special Methods

`HarmonicDiatonicIntervalClassSet.__and__()`

`x.__and__(y) <==> x&y`

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.__cmp__(y) <==> cmp(x, y)`

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.__contains__()`

`x.__contains__(y) <==> y in x.`

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.frozenset`

```

HarmonicDiatonicIntervalClassSet.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__hash__() <==> hash(x)
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__iter__() <==> iter(x)
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__repr__()

HarmonicDiatonicIntervalClassSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

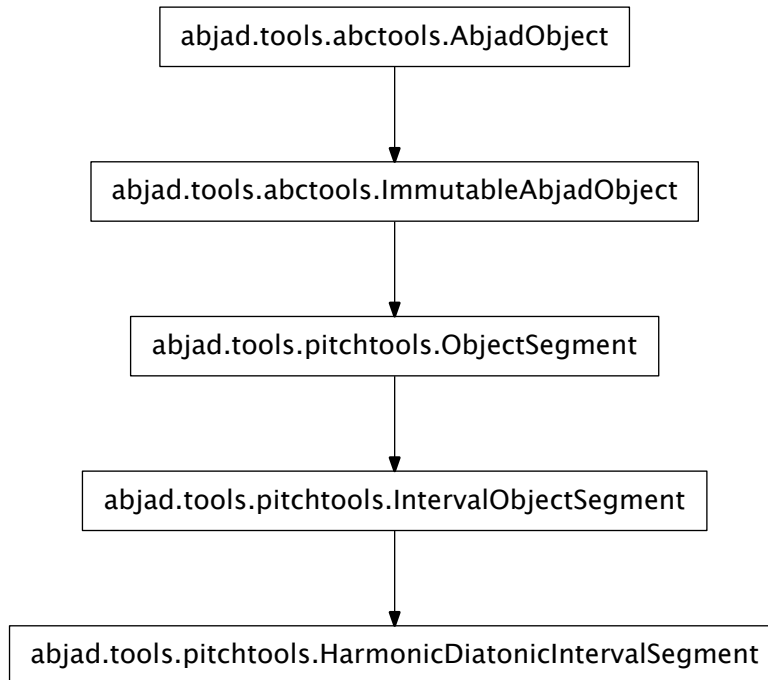
HarmonicDiatonicIntervalClassSet.__str__()

HarmonicDiatonicIntervalClassSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalClassSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset

```

pitchtools.HarmonicDiatonicIntervalSegment



class `pitchtools.HarmonicDiatonicIntervalSegment` (*args, **kwargs)

New in version 2.0. Abjad model of harmonic diatonic interval segment:

```
>>> pitchtools.HarmonicDiatonicIntervalSegment('m2 M9 m3 M3')
HarmonicDiatonicIntervalSegment('m2 M9 m3 M3')
```

Harmonic diatonic interval segments are immutable.

Read-only Properties

`HarmonicDiatonicIntervalSegment.harmonic_chromatic_interval_segment`

`HarmonicDiatonicIntervalSegment.interval_classes`

Inherited from `pitchtools.IntervalObjectSegment`

`HarmonicDiatonicIntervalSegment.intervals`

Inherited from `pitchtools.IntervalObjectSegment`

`HarmonicDiatonicIntervalSegment.melodic_chromatic_interval_segment`

`HarmonicDiatonicIntervalSegment.melodic_diatonic_interval_segment`

`HarmonicDiatonicIntervalSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`HarmonicDiatonicIntervalSegment.count(value) → integer` – return number of occurrences of value

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.index(value[, start[, stop]]) → integer` – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.rotate(n)`

Inherited from `pitchtools.IntervalObjectSegment`

Special Methods

`HarmonicDiatonicIntervalSegment.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`HarmonicDiatonicIntervalSegment.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__getslice__(start, stop)`

Inherited from `pitchtools.ObjectSegment`

`HarmonicDiatonicIntervalSegment.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__hash__() <==> hash(x)`

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__iter__() <==> iter(x)`

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__le__()`

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__len__() <==> len(x)`

Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__lt__()`

`x.__lt__(y) <==> x<y`

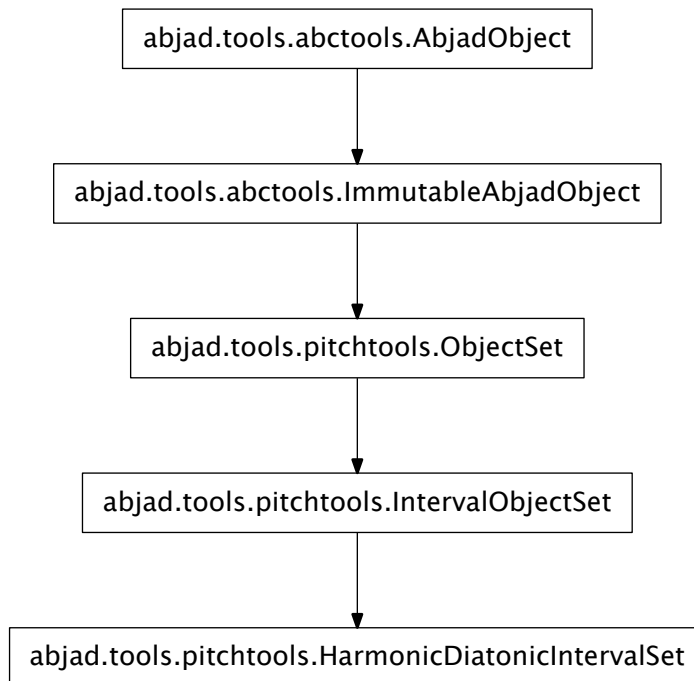
Inherited from `__builtin__.tuple`

```

HarmonicDiatonicIntervalSegment.__mul__(n)
    Inherited from pitchtools.ObjectSegment
HarmonicDiatonicIntervalSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple
HarmonicDiatonicIntervalSegment.__repr__()
HarmonicDiatonicIntervalSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment
HarmonicDiatonicIntervalSegment.__str__()
    Inherited from pitchtools.IntervalObjectSegment

```

pitchtools.HarmonicDiatonicIntervalSet



```

class pitchtools.HarmonicDiatonicIntervalSet(*args, **kwargs)
    New in version 2.0. Abjad model of harmonic diatonic interval set:

    >>> pitchtools.HarmonicDiatonicIntervalSet('m2 m2 M2 M9')
    HarmonicDiatonicIntervalSet('m2 M2 M9')

```

Harmonic diatonic interval sets are immutable.

Read-only Properties

`HarmonicDiatonicIntervalSet.harmonic_chromatic_interval_set`
`HarmonicDiatonicIntervalSet.harmonic_diatonic_interval_numbers`
`HarmonicDiatonicIntervalSet.harmonic_diatonic_intervals`
`HarmonicDiatonicIntervalSet.storage_format`
 Storage format of Abjad object.
 Return string.
 Inherited from `abctools.AbjadObject`

Methods

`HarmonicDiatonicIntervalSet.copy()`
 Return a shallow copy of a set.
 Inherited from `__builtin__.frozenset`
`HarmonicDiatonicIntervalSet.difference()`
 Return the difference of two or more sets as a new set.
 (i.e. all elements that are in this set but not the others.)
 Inherited from `__builtin__.frozenset`
`HarmonicDiatonicIntervalSet.intersection()`
 Return the intersection of two or more sets as a new set.
 (i.e. elements that are common to all of the sets.)
 Inherited from `__builtin__.frozenset`
`HarmonicDiatonicIntervalSet.isdisjoint()`
 Return True if two sets have a null intersection.
 Inherited from `__builtin__.frozenset`
`HarmonicDiatonicIntervalSet.issubset()`
 Report whether another set contains this set.
 Inherited from `__builtin__.frozenset`
`HarmonicDiatonicIntervalSet.issuperset()`
 Report whether this set contains another set.
 Inherited from `__builtin__.frozenset`
`HarmonicDiatonicIntervalSet.symmetric_difference()`
 Return the symmetric difference of two sets as a new set.
 (i.e. all elements that are in exactly one of the sets.)
 Inherited from `__builtin__.frozenset`
`HarmonicDiatonicIntervalSet.union()`
 Return the union of sets as a new set.
 (i.e. all elements that are in either set.)
 Inherited from `__builtin__.frozenset`

Special Methods

```

HarmonicDiatonicIntervalSet.__and__()
    x.__and__(y) <==> x&y

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__cmp__(y) <==> cmp(x, y)
    Inherited from __builtin__.frozenset

HarmonicDiatonicIntervalSet.__contains__()
    x.__contains__(y) <==> y in x.

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__eq__()
    x.__eq__(y) <==> x==y

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__ge__()
    x.__ge__(y) <==> x>=y

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__gt__()
    x.__gt__(y) <==> x>y

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__hash__() <==> hash(x)
    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__iter__() <==> iter(x)
    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__le__()
    x.__le__(y) <==> x<=y

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__lt__()
    x.__lt__(y) <==> x<y

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__ne__()
    x.__ne__(y) <==> x!=y

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__or__()
    x.__or__(y) <==> x|y

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__rand__()
    x.__rand__(y) <==> y&x

    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__repr__()

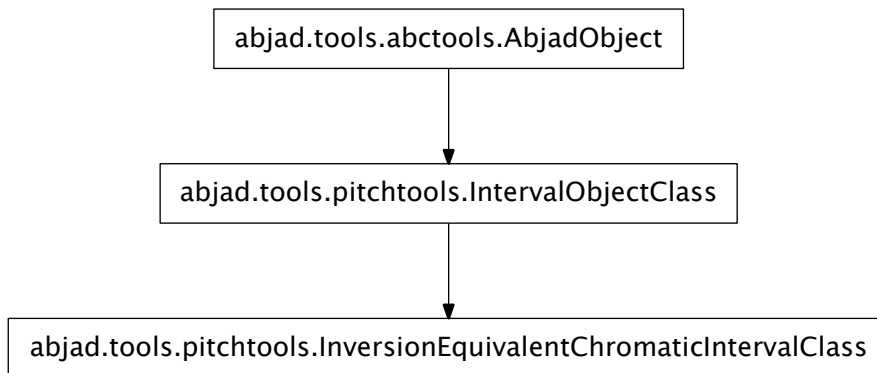
```

```

HarmonicDiatonicIntervalSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__str__()
HarmonicDiatonicIntervalSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset
HarmonicDiatonicIntervalSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset

```

pitchtools.InversionEquivalentChromaticIntervalClass



class `pitchtools.InversionEquivalentChromaticIntervalClass` (*interval_class_token*)
 New in version 2.0. Abjad model of inversion-equivalent chromatic interval-class:

```

>>> pitchtools.InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentChromaticIntervalClass(1)

```

Inversion-equivalent chromatic interval-classes are immutable.

Read-only Properties

`InversionEquivalentChromaticIntervalClass.inversion_equivalent_chromatic_interval_number`

`InversionEquivalentChromaticIntervalClass.number`

Inherited from `pitchtools.IntervalObjectClass`

`InversionEquivalentChromaticIntervalClass.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`InversionEquivalentChromaticIntervalClass.__abs__()`

`InversionEquivalentChromaticIntervalClass.__eq__(arg)`

`InversionEquivalentChromaticIntervalClass.__float__()`

Inherited from `pitchtools.IntervalObjectClass`

`InversionEquivalentChromaticIntervalClass.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InversionEquivalentChromaticIntervalClass.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`InversionEquivalentChromaticIntervalClass.__hash__()`

`InversionEquivalentChromaticIntervalClass.__int__()`

Inherited from `pitchtools.IntervalObjectClass`

`InversionEquivalentChromaticIntervalClass.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InversionEquivalentChromaticIntervalClass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

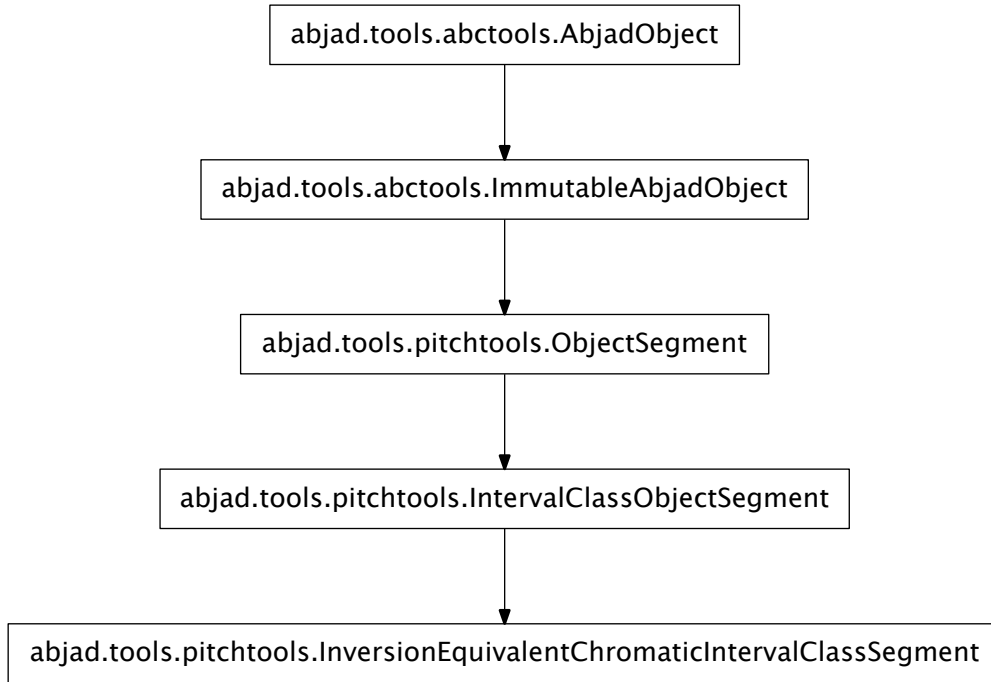
`InversionEquivalentChromaticIntervalClass.__ne__(arg)`

`InversionEquivalentChromaticIntervalClass.__neg__()`

`InversionEquivalentChromaticIntervalClass.__repr__()`

`InversionEquivalentChromaticIntervalClass.__str__()`

pitchtools.InversionEquivalentChromaticIntervalClassSegment



```
class pitchtools.InversionEquivalentChromaticIntervalClassSegment (*args,
                                                                    **kwargs)
```

New in version 2.0. Abjad model of inversion-equivalent chromatic interval-class segment:

```
>>> pitchtools.InversionEquivalentChromaticIntervalClassSegment([2, 1, 0, 5.5, 6])
InversionEquivalentChromaticIntervalClassSegment(2, 1, 0, 5.5, 6)
```

Inversion-equivalent chromatic interval-class segments are immutable.

Read-only Properties

`InversionEquivalentChromaticIntervalClassSegment.interval_class_numbers`

Inherited from `pitchtools.IntervalClassObjectSegment`

`InversionEquivalentChromaticIntervalClassSegment.interval_classes`

Inherited from `pitchtools.IntervalClassObjectSegment`

`InversionEquivalentChromaticIntervalClassSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`InversionEquivalentChromaticIntervalClassSegment.count(value) → integer` – return number of occurrences of value

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.index(value[, start[, stop]]) → integer` – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

Special Methods

`InversionEquivalentChromaticIntervalClassSegment.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`InversionEquivalentChromaticIntervalClassSegment.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__getslice__(start, stop)`

Inherited from `pitchtools.ObjectSegment`

`InversionEquivalentChromaticIntervalClassSegment.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__hash__() <==> hash(x)`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__iter__() <==> iter(x)`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__le__()`

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__len__() <==> len(x)`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__lt__()`

`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__mul__(n)`

Inherited from `pitchtools.ObjectSegment`


```
InversionEquivalentChromaticIntervalClassSegment.__ne__()
```

```
x.__ne__(y) <==> x!=y
```

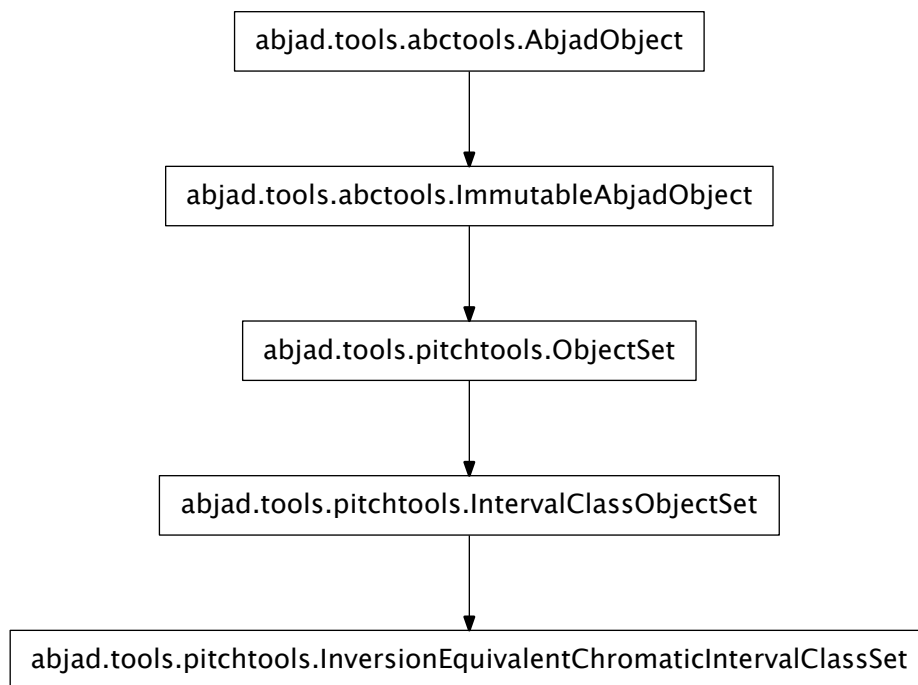
```
Inherited from __builtin__.tuple
```

```
InversionEquivalentChromaticIntervalClassSegment.__repr__()
```

```
InversionEquivalentChromaticIntervalClassSegment.__rmul__(n)
```

```
Inherited from pitchtools.ObjectSegment
```

pitchtools.InversionEquivalentChromaticIntervalClassSet



```
class pitchtools.InversionEquivalentChromaticIntervalClassSet (*args, **kwargs)
```

New in version 2.0. Abjad model of inversion-equivalent chromatic interval-class set:

```
>>> pitchtools.InversionEquivalentChromaticIntervalClassSet([1, 1, 6, 2, 2])
InversionEquivalentChromaticIntervalClassSet(1, 2, 6)
```

Inversion-equivalent chromatic interval-class sets are immutable.

Read-only Properties

```
InversionEquivalentChromaticIntervalClassSet.inversion_equivalent_chromatic_interval_class
```

```
InversionEquivalentChromaticIntervalClassSet.inversion_equivalent_chromatic_interval_class
```

```
InversionEquivalentChromaticIntervalClassSet.storage_format
```

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`InversionEquivalentChromaticIntervalClassSet.copy()`

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.isdisjoint()`

Return True if two sets have a null intersection.

Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.issubset()`

Report whether another set contains this set.

Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.issuperset()`

Report whether this set contains another set.

Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.union()`

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Inherited from `__builtin__.frozenset`

Special Methods

`InversionEquivalentChromaticIntervalClassSet.__and__()`

`x.__and__(y) <==> x&y`

Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__cmp__(y) <==> cmp(x, y)`

Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__contains__()`

`x.__contains__(y) <==> y in x.`

Inherited from `__builtin__.frozenset`

```

InversionEquivalentChromaticIntervalClassSet.__eq__()
    x.__eq__(y) <==> x==y
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__hash__() <==> hash(x)
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__iter__() <==> iter(x)
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__repr__()

InversionEquivalentChromaticIntervalClassSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

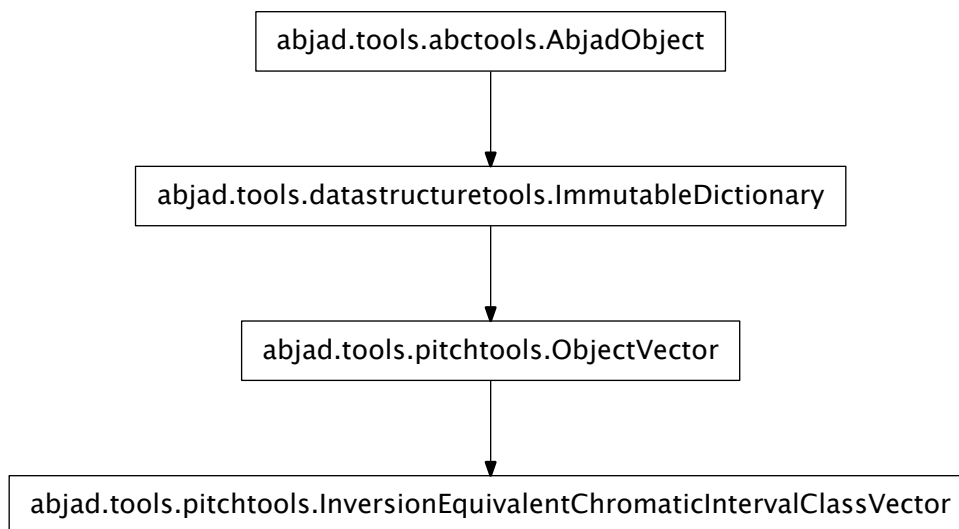
InversionEquivalentChromaticIntervalClassSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

```

```
InversionEquivalentChromaticIntervalClassSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

InversionEquivalentChromaticIntervalClassSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset
```

pitchtools.InversionEquivalentChromaticIntervalClassVector



```
class pitchtools.InversionEquivalentChromaticIntervalClassVector(*args,
                                                                **kwargs)
```

New in version 2.0. Abjad model of inversion-equivalent chromatic interval-class vector:

```
>>> pitchtools.InversionEquivalentChromaticIntervalClassVector([1, 1, 6, 2, 2, 2])
InversionEquivalentChromaticIntervalClassVector(0 | 2 3 0 0 0 1)
```

Initialize by inversion-equivalent chromatic interval-class counts:

```
>>> pitchtools.InversionEquivalentChromaticIntervalClassVector(counts=[2, 3, 0, 0, 0, 1])
InversionEquivalentChromaticIntervalClassVector(0 | 2 3 0 0 0 1)
```

Inversion-equivalent chromatic interval-class vectors are immutable.

Read-only Properties

`InversionEquivalentChromaticIntervalClassVector.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`InversionEquivalentChromaticIntervalClassVector.clear()` → None. Remove all items from D.

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.copy()` → a shallow copy of D

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.has_key(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.items()` → list of D's (key, value) pairs, as 2-tuples

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.iteritems()` → an iterator over the (key, value) items of D

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.iterkeys()` → an iterator over the keys of D

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.itervalues()` → an iterator over the values of D

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.keys()` → list of D's keys

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.pop(k[, d])` → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.popitem()` → (k, v), remove and return some (key, value) pair as a

2-tuple; but raise `KeyError` if D is empty.

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.update([E], **F)` → None. Update D from dict/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.values()` → list of D's values

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.viewitems()` → a set-like object providing a view on D's items

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.viewkeys()` → a set-like object providing a view on D's keys

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.viewvalues()` → an object providing a view on D's values

Inherited from `__builtin__.dict`

Special Methods

`InversionEquivalentChromaticIntervalClassVector.__cmp__(y) <==> cmp(x, y)`

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__contains__(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__delitem__(*args)`

Inherited from `datastructuretools.ImmutableDictionary`

`InversionEquivalentChromaticIntervalClassVector.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__iter__()` <==> `iter(x)`

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__le__()`

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__len__()` <==> `len(x)`

Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__lt__()`

`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.dict`

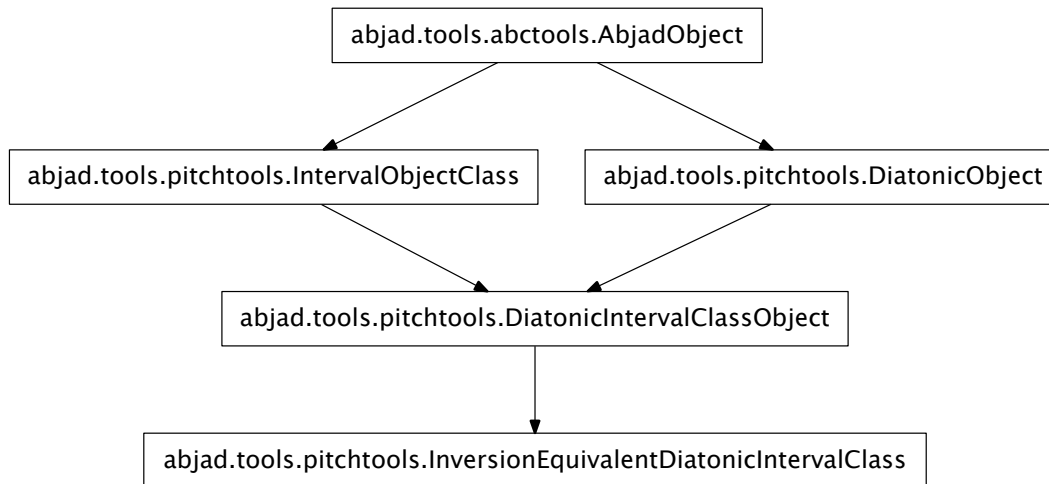
`InversionEquivalentChromaticIntervalClassVector.__ne__()`

`x.__ne__(y) <==> x!=y`

Inherited from `__builtin__.dict`

```
InversionEquivalentChromaticIntervalClassVector.__repr__()
InversionEquivalentChromaticIntervalClassVector.__setitem__(*args)
    Inherited from datastructuretools.ImmutableDictionary
```

pitchtools.InversionEquivalentDiatonicIntervalClass



```
class pitchtools.InversionEquivalentDiatonicIntervalClass(*args)
    New in version 2.0. Abjad model of inversion-equivalent diatonic interval-class:

    >>> pitchtools.InversionEquivalentDiatonicIntervalClass('-m14')
    InversionEquivalentDiatonicIntervalClass('M2')
```

Inversion-equivalent diatonic interval-classes are immutable.

Read-only Properties

```
InversionEquivalentDiatonicIntervalClass.number
    Inherited from pitchtools.IntervalObjectClass
InversionEquivalentDiatonicIntervalClass.quality_string
    Inherited from pitchtools.DiatonicIntervalClassObject
InversionEquivalentDiatonicIntervalClass.storage_format
    Storage format of Abjad object.

    Return string.

    Inherited from abctools.AbjadObject
```

Special Methods

```
InversionEquivalentDiatonicIntervalClass.__abs__()
    Inherited from pitchtools.DiatonicIntervalClassObject
InversionEquivalentDiatonicIntervalClass.__eq__(arg)
```

`InversionEquivalentDiatonicIntervalClass.__float__()`
Inherited from `pitchtools.DiatonicIntervalClassObject`

`InversionEquivalentDiatonicIntervalClass.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InversionEquivalentDiatonicIntervalClass.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`InversionEquivalentDiatonicIntervalClass.__hash__()`
Inherited from `pitchtools.IntervalObjectClass`

`InversionEquivalentDiatonicIntervalClass.__int__()`
Inherited from `pitchtools.DiatonicIntervalClassObject`

`InversionEquivalentDiatonicIntervalClass.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InversionEquivalentDiatonicIntervalClass.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

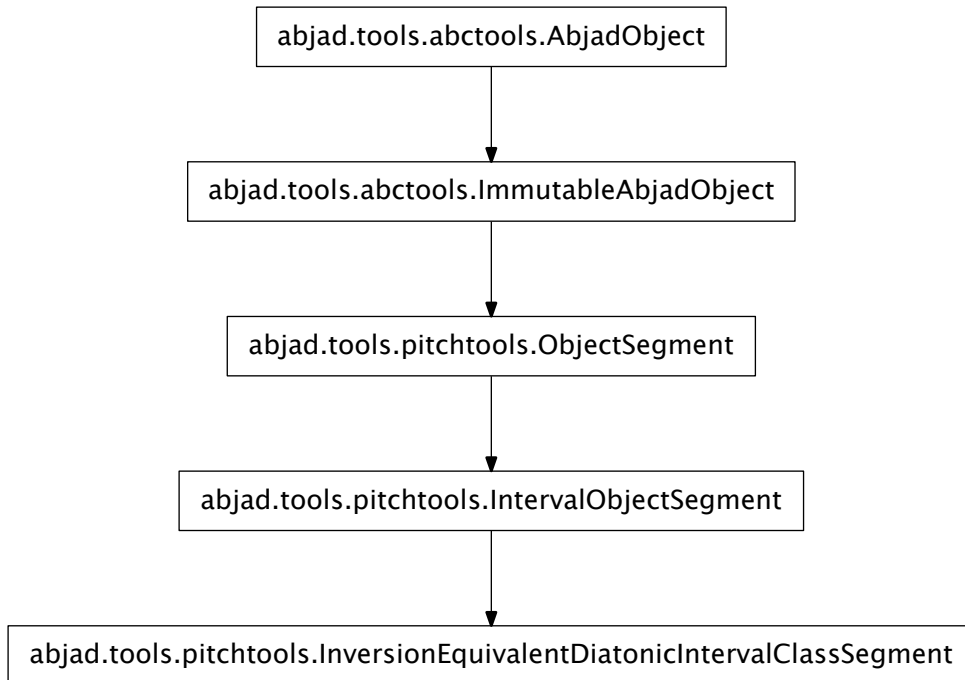
Inherited from `abctools.AbjadObject`

`InversionEquivalentDiatonicIntervalClass.__ne__(arg)`

`InversionEquivalentDiatonicIntervalClass.__repr__()`
Inherited from `pitchtools.DiatonicIntervalClassObject`

`InversionEquivalentDiatonicIntervalClass.__str__()`
Inherited from `pitchtools.DiatonicIntervalClassObject`

pitchtools.InversionEquivalentDiatonicIntervalClassSegment



```
class pitchtools.InversionEquivalentDiatonicIntervalClassSegment (*args,
                                                                **kwargs)
```

New in version 2.0. Abjad model of inversion-equivalent diatonic interval-class segment:

```
>>> pitchtools.InversionEquivalentDiatonicIntervalClassSegment (
... [ ('major', 2), ('major', 9), ('minor', -2), ('minor', -9)])
InversionEquivalentDiatonicIntervalClassSegment (M2, M2, m2, m2)
```

Inversion-equivalent diatonic interval-class segments are immutable.

Read-only Properties

`InversionEquivalentDiatonicIntervalClassSegment.interval_classes`

Inherited from `pitchtools.IntervalObjectSegment`

`InversionEquivalentDiatonicIntervalClassSegment.intervals`

Inherited from `pitchtools.IntervalObjectSegment`

`InversionEquivalentDiatonicIntervalClassSegment.is_tertian`

True when all diatonic interval-classes in segment are tertian. Otherwise false:

```
>>> dics = pitchtools.InversionEquivalentDiatonicIntervalClassSegment (
... [ ('major', 3), ('minor', 6), ('major', 6)])
>>> dics.is_tertian
True
```

Return boolean.

`InversionEquivalentDiatonicIntervalClassSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`InversionEquivalentDiatonicIntervalClassSegment.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.rotate(n)`

Inherited from `pitchtools.IntervalObjectSegment`

Special Methods

`InversionEquivalentDiatonicIntervalClassSegment.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`InversionEquivalentDiatonicIntervalClassSegment.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__getslice__(start, stop)`

Inherited from `pitchtools.ObjectSegment`

`InversionEquivalentDiatonicIntervalClassSegment.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__hash__() <==> hash(x)`

Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__iter__() <==> iter(x)`

Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__le__()`

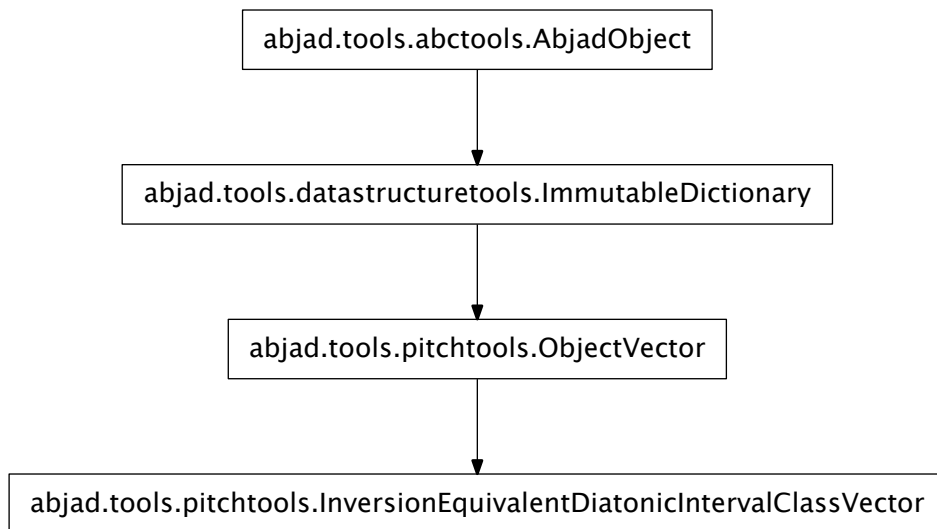
`x.__le__(y) <==> x<=y`

```

    Inherited from __builtin__.tuple
InversionEquivalentDiatonicIntervalClassSegment.__len__() <==> len(x)
    Inherited from __builtin__.tuple
InversionEquivalentDiatonicIntervalClassSegment.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple
InversionEquivalentDiatonicIntervalClassSegment.__mul__(n)
    Inherited from pitchtools.ObjectSegment
InversionEquivalentDiatonicIntervalClassSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple
InversionEquivalentDiatonicIntervalClassSegment.__repr__()
    Inherited from pitchtools.IntervalObjectSegment
InversionEquivalentDiatonicIntervalClassSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment
InversionEquivalentDiatonicIntervalClassSegment.__str__()
    Inherited from pitchtools.IntervalObjectSegment

```

pitchtools.InversionEquivalentDiatonicIntervalClassVector



class pitchtools.**InversionEquivalentDiatonicIntervalClassVector** (*expr*)

New in version 2.0. Abjad model of inversion-equivalent diatonic interval-class vector:

```

>>> staff = Staff("c'8 d'8 e'8 f'8 g'8")
>>> vector = pitchtools.InversionEquivalentDiatonicIntervalClassVector(staff)

```

```
>>> print vector
{P1: 0, aug1: 0, m2: 1, M2: 3, aug2: 0, dim3: 0, m3: 2, M3: 1, dim4: 0, P4: 3, aug4: 0}
```

Inversion-equivalent diatonic interval-class vector are not quartertone-aware.

Inversion-equivalent diatonic interval-class vectors are immutable.

Read-only Properties

`InversionEquivalentDiatonicIntervalClassVector.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`InversionEquivalentDiatonicIntervalClassVector.clear()` → None. Remove all items from D.

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.copy()` → a shallow copy of D

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.has_key(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.items()` → list of D's (key, value) pairs, as 2-tuples

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.iteritems()` → an iterator over the (key, value) items of D

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.iterkeys()` → an iterator over the keys of D

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.itervalues()` → an iterator over the values of D

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.keys()` → list of D's keys

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.pop(k[, d])` → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.popitem()` → (k, v), remove and return some (key, value) pair as a

2-tuple; but raise `KeyError` if D is empty.

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.setdefault(k[, d])` → `D.get(k,d)`,
also set `D[k]=d` if `k` not
in `D`

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.update([E], **F)` → `None`. Update
`D` from dict/iterable `E` and `F`.
If `E` present and has a `.keys()` method, does: for `k` in `E`: `D[k] = E[k]` If `E` present and lacks `.keys()` method, does:
for `(k, v)` in `E`: `D[k] = v` In either case, this is followed by: for `k` in `F`: `D[k] = F[k]`

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.values()` → list of `D`'s values

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.viewitems()` → a set-like object
providing a view on `D`'s
items

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.viewkeys()` → a set-like object pro-
viding a view on `D`'s keys

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.viewvalues()` → an object providing
a view on `D`'s values

Inherited from `__builtin__.dict`

Special Methods

`InversionEquivalentDiatonicIntervalClassVector.__cmp__(y)` <==> `cmp(x, y)`

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__contains__(k)` → True if `D` has a
key `k`, else False

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__delitem__(*args)`

Inherited from `datastructuretools.ImmutableDictionary`

`InversionEquivalentDiatonicIntervalClassVector.__eq__(arg)`

`InversionEquivalentDiatonicIntervalClassVector.__ge__()`

`x.__ge__(y)` <==> `x>=y`

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__getitem__()`

`x.__getitem__(y)` <==> `x[y]`

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__gt__()`

`x.__gt__(y)` <==> `x>y`

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__iter__()` <==> `iter(x)`

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__le__()`

`x.__le__(y)` <==> `x<=y`

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__len__()` $\iff \text{len}(x)$

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__lt__()`

`x.__lt__(y) \iff x < y`

Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__ne__(arg)`

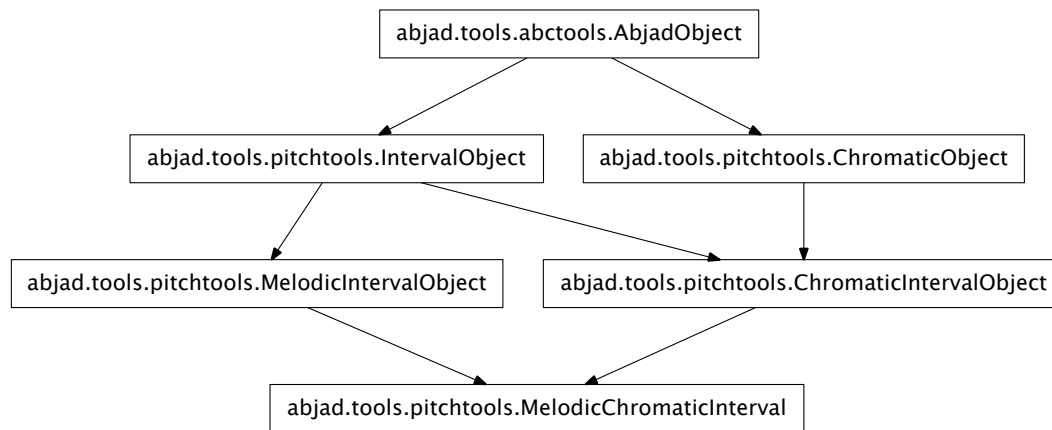
`InversionEquivalentDiatonicIntervalClassVector.__repr__()`

`InversionEquivalentDiatonicIntervalClassVector.__setitem__(*args)`

Inherited from `datastructuretools.ImmutableDictionary`

`InversionEquivalentDiatonicIntervalClassVector.__str__()`

`pitchtools.MelodicChromaticInterval`



class `pitchtools.MelodicChromaticInterval` (*arg*)

New in version 2.0. Abjad model of melodic chromatic interval:

```
>>> pitchtools.MelodicChromaticInterval(-14)
MelodicChromaticInterval(-14)
```

Melodic chromatic intervals are immutable.

Read-only Properties

`MelodicChromaticInterval.cents`

Inherited from `pitchtools.IntervalObject`

`MelodicChromaticInterval.chromatic_interval_number`

Read-only chromatic interval number:

```
>>> pitchtools.MelodicChromaticInterval(-14).chromatic_interval_number
-14
```

Return integer or float.

`MelodicChromaticInterval.direction_number`

Read-only numeric sign:

```
>>> pitchtools.MelodicChromaticInterval(-14).direction_number
-1
```

Return integer.

`MelodicChromaticInterval.direction_string`

Inherited from `pitchtools.MelodicIntervalObject`

`MelodicChromaticInterval.harmonic_chromatic_interval`

Read-only harmonic chromatic interval:

```
>>> pitchtools.MelodicChromaticInterval(-14).harmonic_chromatic_interval
HarmonicChromaticInterval(14)
```

Return harmonic chromatic interval.

`MelodicChromaticInterval.interval_class`

Inherited from `pitchtools.IntervalObject`

`MelodicChromaticInterval.melodic_chromatic_interval_class`

Read-only melodic chromatic interval-class:

```
>>> pitchtools.MelodicChromaticInterval(-14).melodic_chromatic_interval_class
MelodicChromaticIntervalClass(-2)
```

Return melodic chromatic interval-class.

`MelodicChromaticInterval.number`

Inherited from `pitchtools.ChromaticIntervalObject`

`MelodicChromaticInterval.semitones`

Inherited from `pitchtools.ChromaticIntervalObject`

`MelodicChromaticInterval.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`MelodicChromaticInterval.__abs__()`

`MelodicChromaticInterval.__add__(arg)`

Inherited from `pitchtools.ChromaticIntervalObject`

`MelodicChromaticInterval.__eq__(arg)`

Inherited from `pitchtools.ChromaticIntervalObject`

`MelodicChromaticInterval.__float__()`

Inherited from `pitchtools.ChromaticIntervalObject`

`MelodicChromaticInterval.__ge__(arg)`

`MelodicChromaticInterval.__gt__(arg)`

`MelodicChromaticInterval.__hash__()`

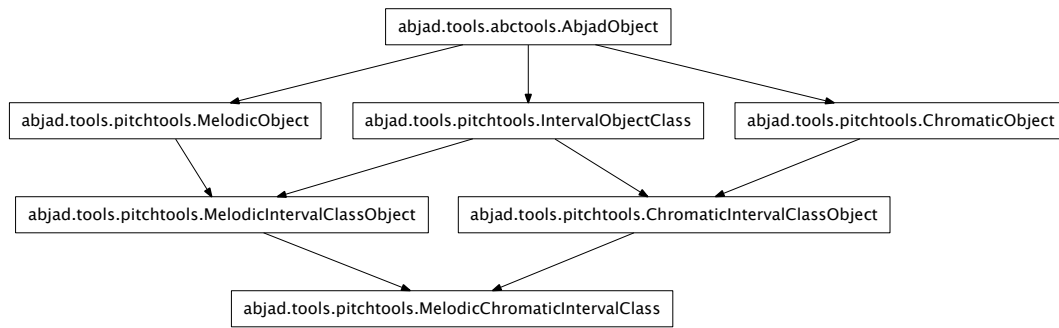
`MelodicChromaticInterval.__int__()`

Inherited from `pitchtools.ChromaticIntervalObject`

`MelodicChromaticInterval.__le__(arg)`

```
MelodicChromaticInterval.__lt__(arg)
MelodicChromaticInterval.__ne__(arg)
    Inherited from pitchtools.ChromaticIntervalObject
MelodicChromaticInterval.__neg__()
MelodicChromaticInterval.__repr__()
MelodicChromaticInterval.__str__()
MelodicChromaticInterval.__sub__(arg)
    Inherited from pitchtools.ChromaticIntervalObject
```

pitchtools.MelodicChromaticIntervalClass



class `pitchtools.MelodicChromaticIntervalClass` (*token*)
 New in version 2.0. Abjad model of melodic chromatic interval-class:

```
>>> pitchtools.MelodicChromaticIntervalClass(-14)
MelodicChromaticIntervalClass(-2)
```

Melodic chromatic interval-classes are immutable.

Read-only Properties

```
MelodicChromaticIntervalClass.direction_number
    Inherited from pitchtools.MelodicIntervalClassObject
MelodicChromaticIntervalClass.direction_symbol
    Inherited from pitchtools.MelodicIntervalClassObject
MelodicChromaticIntervalClass.direction_word
    Inherited from pitchtools.MelodicIntervalClassObject
MelodicChromaticIntervalClass.number
    Inherited from pitchtools.IntervalObjectClass
MelodicChromaticIntervalClass.storage_format
    Storage format of Abjad object.

    Return string.

    Inherited from abctools.AbjadObject
```


Special Methods

`MelodicChromaticIntervalClass.__abs__()`
 Inherited from `pitchtools.ChromaticIntervalClassObject`

`MelodicChromaticIntervalClass.__eq__(arg)`

`MelodicChromaticIntervalClass.__float__()`
 Inherited from `pitchtools.ChromaticIntervalClassObject`

`MelodicChromaticIntervalClass.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`MelodicChromaticIntervalClass.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`MelodicChromaticIntervalClass.__hash__()`
 Inherited from `pitchtools.IntervalObjectClass`

`MelodicChromaticIntervalClass.__int__()`
 Inherited from `pitchtools.ChromaticIntervalClassObject`

`MelodicChromaticIntervalClass.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

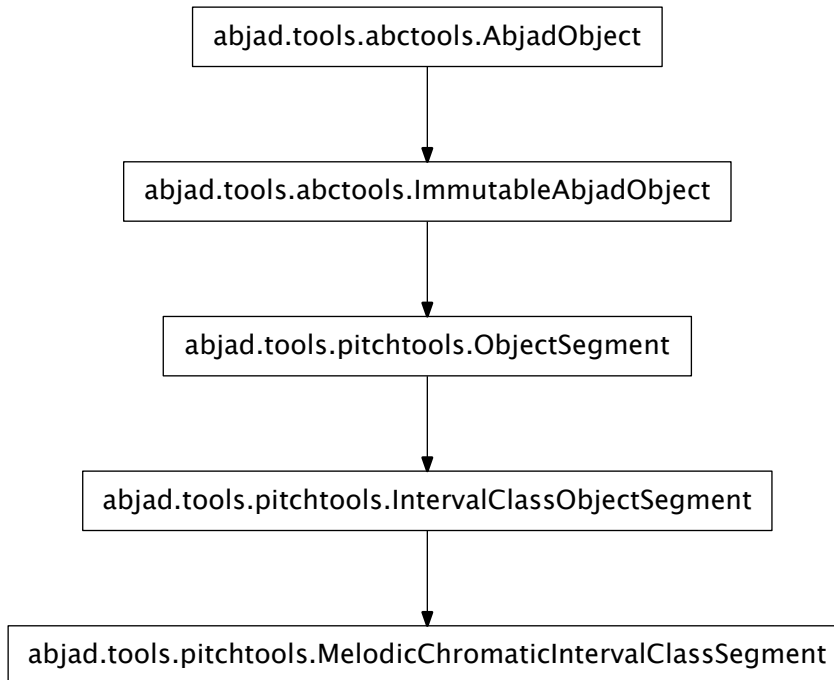
`MelodicChromaticIntervalClass.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`MelodicChromaticIntervalClass.__ne__(arg)`

`MelodicChromaticIntervalClass.__repr__()`
 Inherited from `pitchtools.IntervalObjectClass`

`MelodicChromaticIntervalClass.__str__()`
 Inherited from `pitchtools.IntervalObjectClass`

`pitchtools.MelodicChromaticIntervalClassSegment`



class `pitchtools.MelodicChromaticIntervalClassSegment` (*args, **kwargs)

New in version 2.0. Abjad model of melodic chromatic interval-class segment:

```
>>> pitchtools.MelodicChromaticIntervalClassSegment([-2, -14, 3, 5.5, 6.5])
MelodicChromaticIntervalClassSegment(-2, -2, +3, +5.5, +6.5)
```

Melodic chromatic interval-class segments are immutable.

Read-only Properties

`MelodicChromaticIntervalClassSegment.interval_class_numbers`

Inherited from `pitchtools.IntervalClassObjectSegment`

`MelodicChromaticIntervalClassSegment.interval_classes`

Inherited from `pitchtools.IntervalClassObjectSegment`

`MelodicChromaticIntervalClassSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

MelodicChromaticIntervalClassSegment.**count** (*value*) → integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**index** (*value*[, *start*[, *stop*]]) → integer – return first index of value.

Raises ValueError if the value is not present.

Inherited from `__builtin__.tuple`

Special Methods

MelodicChromaticIntervalClassSegment.**__add__** (*arg*)

Inherited from `pitchtools.ObjectSegment`

MelodicChromaticIntervalClassSegment.**__contains__** ()

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__eq__** ()

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__ge__** ()

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__getitem__** ()

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__getslice__** (*start*, *stop*)

Inherited from `pitchtools.ObjectSegment`

MelodicChromaticIntervalClassSegment.**__gt__** ()

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__hash__** () <==> *hash(x)*

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__iter__** () <==> *iter(x)*

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__le__** ()

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__len__** () <==> *len(x)*

Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__lt__** ()

`x.__lt__(y) <==> x<y`

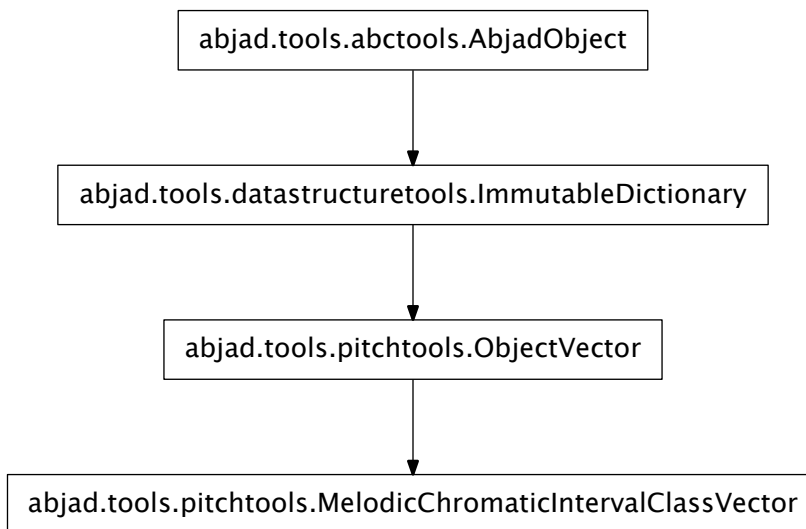
Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.**__mul__** (*n*)

Inherited from `pitchtools.ObjectSegment`

```
MelodicChromaticIntervalClassSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple
MelodicChromaticIntervalClassSegment.__repr__()
    Inherited from pitchtools.IntervalClassObjectSegment
MelodicChromaticIntervalClassSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment
```

pitchtools.MelodicChromaticIntervalClassVector



class pitchtools.**MelodicChromaticIntervalClassVector**(*mcic_tokens*)

New in version 2.0. Abjad model of melodic chromatic interval-class vector:

```
>>> print pitchtools.MelodicChromaticIntervalClassVector([-2, -14, 3, 5.5, 6.5])
. | . . 1 . . . | . . . . .
. | . 2 . . . . | . . . . .
  | . . . . . 1 | 1 . . . .
  | . . . . . . | . . . . .
```

Melodic chromatic interval-class vectors are immutable.

Read-only Properties

`MelodicChromaticIntervalClassVector.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`MelodicChromaticIntervalClassVector.clear()` → None. Remove all items from D.

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.copy()` → a shallow copy of D

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.has_key(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.items()` → list of D's (key, value) pairs, as 2-tuples

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.iteritems()` → an iterator over the (key, value) items of D

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.iterkeys()` → an iterator over the keys of D

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.itervalues()` → an iterator over the values of D

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.keys()` → list of D's keys

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.update([E], **F)` → None. Update D from dict/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.values()` → list of D's values

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.viewitems()` → a set-like object providing a view on D's items

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.viewkeys()` → a set-like object providing a view on D's keys

Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.viewvalues()` → an object providing a view on D's values

Inherited from `__builtin__.dict`

Special Methods

MelodicChromaticIntervalClassVector.__cmp__(y) <==> *cmp*(x, y)

Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__contains__(k) → True if D has a key k, else False

Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__delitem__(*args)

Inherited from `datastructuretools.ImmutableDictionary`

MelodicChromaticIntervalClassVector.__eq__()

x.__eq__(y) <==> x==y

Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__ge__()

x.__ge__(y) <==> x>=y

Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__gt__()

x.__gt__(y) <==> x>y

Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__iter__() <==> *iter*(x)

Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__le__()

x.__le__(y) <==> x<=y

Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__len__()

MelodicChromaticIntervalClassVector.__lt__()

x.__lt__(y) <==> x<y

Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__ne__()

x.__ne__(y) <==> x!=y

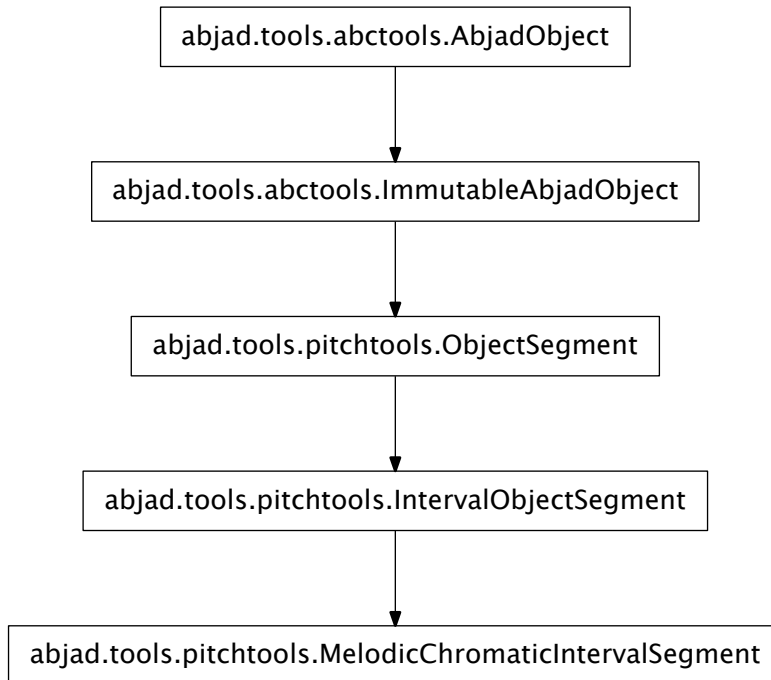
Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__repr__()

MelodicChromaticIntervalClassVector.__setitem__(*args)

Inherited from `datastructuretools.ImmutableDictionary`

MelodicChromaticIntervalClassVector.__str__()

pitchtools.MelodicChromaticIntervalSegment

class `pitchtools.MelodicChromaticIntervalSegment` (*args, **kwargs)

New in version 2.0. Abjad model of melodic chromatic interval segment:

```
>>> pitchtools.MelodicChromaticIntervalSegment([11, 13, 13.5, -2, 2.5])
MelodicChromaticIntervalSegment(+11, +13, +13.5, -2, +2.5)
```

Melodic chromatic interval segments are immutable.

Read-only Properties

`MelodicChromaticIntervalSegment.harmonic_chromatic_interval_segment`

`MelodicChromaticIntervalSegment.interval_classes`

Inherited from `pitchtools.IntervalObjectSegment`

`MelodicChromaticIntervalSegment.intervals`

Inherited from `pitchtools.IntervalObjectSegment`

`MelodicChromaticIntervalSegment.melodic_chromatic_interval_class_segment`

`MelodicChromaticIntervalSegment.melodic_chromatic_interval_class_vector`

`MelodicChromaticIntervalSegment.melodic_chromatic_interval_numbers`

`MelodicChromaticIntervalSegment.slope`

The slope of a melodic interval segment is the sum of its intervals divided by its length:

```
>>> pitchtools.MelodicChromaticIntervalSegment([1, 2]).slope
Fraction(3, 2)
```

Return fraction.

`MelodicChromaticIntervalSegment.spread`

The maximum harmonic interval spanned by any combination of the intervals within a harmonic chromatic interval segment:

```
>>> pitchtools.MelodicChromaticIntervalSegment([1, 2, -3, 1, -2, 1]).spread
HarmonicChromaticInterval(4)
>>> pitchtools.MelodicChromaticIntervalSegment([1, 1, 1, 2, -3, -2]).spread
HarmonicChromaticInterval(5)
```

Return harmonic chromatic interval.

`MelodicChromaticIntervalSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`MelodicChromaticIntervalSegment.count(value)` \rightarrow integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.index(value[, start[, stop]])` \rightarrow integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.rotate(n)`

Inherited from `pitchtools.IntervalObjectSegment`

Special Methods

`MelodicChromaticIntervalSegment.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`MelodicChromaticIntervalSegment.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.__getslice__(start, stop)`

Inherited from `pitchtools.ObjectSegment`


```
MelodicChromaticIntervalSegment.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.tuple

MelodicChromaticIntervalSegment.__hash__() <==> hash(x)
    Inherited from __builtin__.tuple

MelodicChromaticIntervalSegment.__iter__() <==> iter(x)
    Inherited from __builtin__.tuple

MelodicChromaticIntervalSegment.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.tuple

MelodicChromaticIntervalSegment.__len__() <==> len(x)
    Inherited from __builtin__.tuple

MelodicChromaticIntervalSegment.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple

MelodicChromaticIntervalSegment.__mul__(n)
    Inherited from pitchtools.ObjectSegment

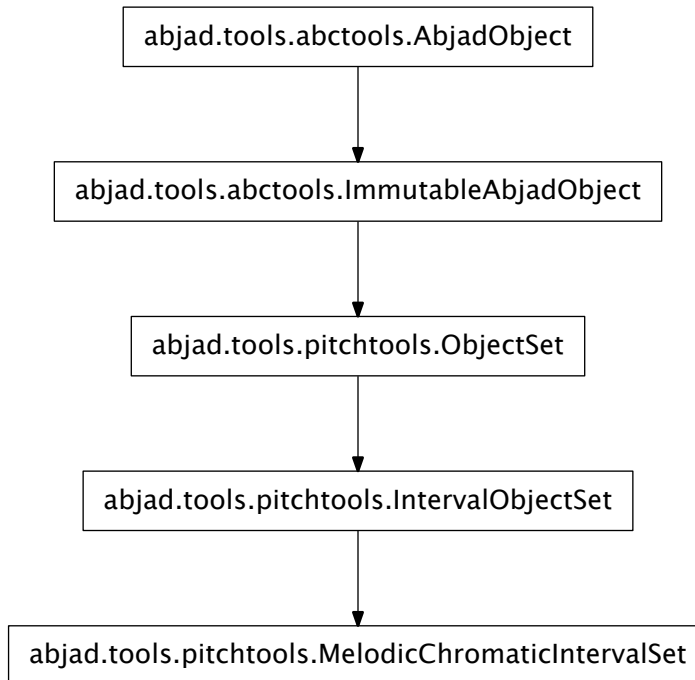
MelodicChromaticIntervalSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple

MelodicChromaticIntervalSegment.__repr__()
    Inherited from pitchtools.IntervalObjectSegment

MelodicChromaticIntervalSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment

MelodicChromaticIntervalSegment.__str__()
    Inherited from pitchtools.IntervalObjectSegment
```

`pitchtools.MelodicChromaticIntervalSet`



class `pitchtools.MelodicChromaticIntervalSet` (*args, **kwargs)

New in version 2.0. Abjad model of melodic chromatic interval set:

```
>>> pitchtools.MelodicChromaticIntervalSet([11, 11, 13.5, 13.5])
MelodicChromaticIntervalSet(+11, +13.5)
```

Melodic chromatic interval sets are immutable.

Read-only Properties

`MelodicChromaticIntervalSet.harmonic_chromatic_interval_set`

`MelodicChromaticIntervalSet.melodic_chromatic_interval_numbers`

`MelodicChromaticIntervalSet.melodic_chromatic_intervals`

`MelodicChromaticIntervalSet.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`MelodicChromaticIntervalSet.copy()`

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.difference()`
 Return the difference of two or more sets as a new set.
 (i.e. all elements that are in this set but not the others.)
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.intersection()`
 Return the intersection of two or more sets as a new set.
 (i.e. elements that are common to all of the sets.)
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.isdisjoint()`
 Return True if two sets have a null intersection.
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.issubset()`
 Report whether another set contains this set.
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.issuperset()`
 Report whether this set contains another set.
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.symmetric_difference()`
 Return the symmetric difference of two sets as a new set.
 (i.e. all elements that are in exactly one of the sets.)
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.union()`
 Return the union of sets as a new set.
 (i.e. all elements that are in either set.)
 Inherited from `__builtin__.frozenset`

Special Methods

`MelodicChromaticIntervalSet.__and__()`
`x.__and__(y) <==> x&y`
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.__cmp__(y) <==> cmp(x, y)`
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.__contains__()`
`x.__contains__(y) <==> y in x.`
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.__eq__()`
`x.__eq__(y) <==> x==y`
 Inherited from `__builtin__.frozenset`
`MelodicChromaticIntervalSet.__ge__()`
`x.__ge__(y) <==> x>=y`

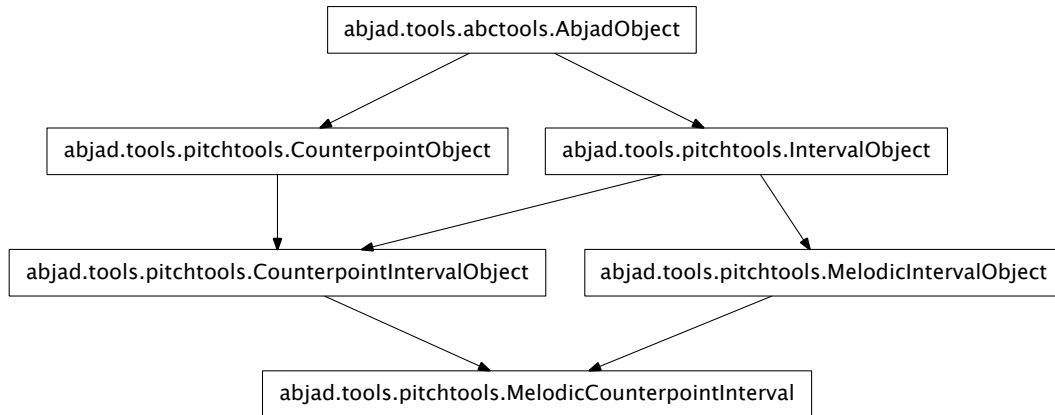
```

    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__hash__() <==> hash(x)
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__iter__() <==> iter(x)
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__repr__()
MelodicChromaticIntervalSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset
MelodicChromaticIntervalSet.__str__()
MelodicChromaticIntervalSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

```

```
MelodicChromaticIntervalSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset
```

pitchtools.MelodicCounterpointInterval



```
class pitchtools.MelodicCounterpointInterval(number)
    New in version 2.0. Abjad model of melodic counterpoint interval:
```

```
>>> pitchtools.MelodicCounterpointInterval(-9)
MelodicCounterpointInterval(-9)
```

Melodic counterpoint intervals are immutable.

Read-only Properties

`MelodicCounterpointInterval.cents`

Inherited from `pitchtools.IntervalObject`

`MelodicCounterpointInterval.direction_number`

`MelodicCounterpointInterval.direction_string`

Inherited from `pitchtools.MelodicIntervalObject`

`MelodicCounterpointInterval.interval_class`

Inherited from `pitchtools.IntervalObject`

`MelodicCounterpointInterval.melodic_counterpoint_interval_class`

`MelodicCounterpointInterval.number`

Inherited from `pitchtools.CounterpointIntervalObject`

`MelodicCounterpointInterval.semitones`

Inherited from `pitchtools.CounterpointIntervalObject`

`MelodicCounterpointInterval.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`MelodicCounterpointInterval.__abs__()`

Inherited from `pitchtools.CounterpointIntervalObject`

`MelodicCounterpointInterval.__eq__(arg)`

Inherited from `pitchtools.MelodicIntervalObject`

`MelodicCounterpointInterval.__float__()`

Inherited from `pitchtools.CounterpointIntervalObject`

`MelodicCounterpointInterval.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicCounterpointInterval.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MelodicCounterpointInterval.__hash__()`

Inherited from `pitchtools.IntervalObject`

`MelodicCounterpointInterval.__int__()`

Inherited from `pitchtools.CounterpointIntervalObject`

`MelodicCounterpointInterval.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicCounterpointInterval.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicCounterpointInterval.__ne__(arg)`

Inherited from `pitchtools.MelodicIntervalObject`

`MelodicCounterpointInterval.__neg__()`

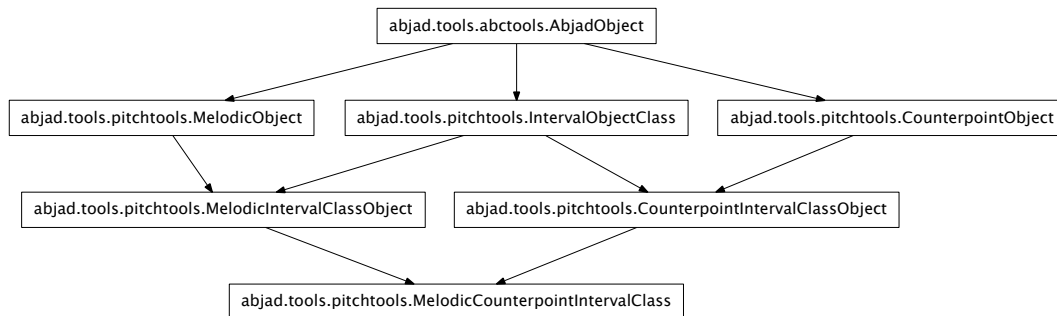
Inherited from `pitchtools.MelodicIntervalObject`

`MelodicCounterpointInterval.__repr__()`

Inherited from `pitchtools.IntervalObject`

`MelodicCounterpointInterval.__str__()`

pitchtools.MelodicCounterpointIntervalClass



class `pitchtools.MelodicCounterpointIntervalClass` (*token*)
 New in version 2.0. Abjad model of melodic counterpoint interval-class:

```
>>> pitchtools.MelodicCounterpointIntervalClass(-9)
MelodicCounterpointIntervalClass(-2)
```

Melodic counterpoint interval-classes are immutable.

Read-only Properties

`MelodicCounterpointIntervalClass.direction_number`
 Inherited from `pitchtools.MelodicIntervalClassObject`

`MelodicCounterpointIntervalClass.direction_symbol`
 Inherited from `pitchtools.MelodicIntervalClassObject`

`MelodicCounterpointIntervalClass.direction_word`
 Inherited from `pitchtools.MelodicIntervalClassObject`

`MelodicCounterpointIntervalClass.number`
 Inherited from `pitchtools.IntervalObjectClass`

`MelodicCounterpointIntervalClass.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`MelodicCounterpointIntervalClass.__abs__()`
 Inherited from `pitchtools.CounterpointIntervalClassObject`

`MelodicCounterpointIntervalClass.__eq__(arg)`

`MelodicCounterpointIntervalClass.__float__()`
 Inherited from `pitchtools.CounterpointIntervalClassObject`

`MelodicCounterpointIntervalClass.__ge__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicCounterpointIntervalClass.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MelodicCounterpointIntervalClass.__hash__()`

Inherited from `pitchtools.IntervalObjectClass`

`MelodicCounterpointIntervalClass.__int__()`

Inherited from `pitchtools.CounterpointIntervalClassObject`

`MelodicCounterpointIntervalClass.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicCounterpointIntervalClass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicCounterpointIntervalClass.__ne__(arg)`

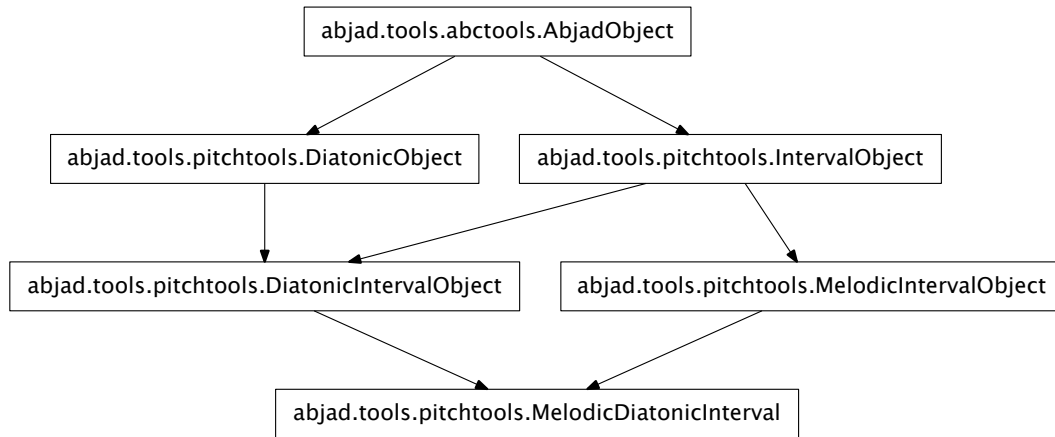
`MelodicCounterpointIntervalClass.__repr__()`

Inherited from `pitchtools.IntervalObjectClass`

`MelodicCounterpointIntervalClass.__str__()`

Inherited from `pitchtools.IntervalObjectClass`

pitchtools.MelodicDiatonicInterval



class `pitchtools.MelodicDiatonicInterval` (*args)
 New in version 2.0. Abjad model of melodic diatonic interval:

```
>>> pitchtools.MelodicDiatonicInterval('+M9')
MelodicDiatonicInterval('+M9')
```

Melodic diatonic intervals are immutable.

Read-only Properties

`MelodicDiatonicInterval.cents`

Inherited from `pitchtools.IntervalObject`

`MelodicDiatonicInterval.diatonic_interval_class`

Inherited from `pitchtools.DiatonicIntervalObject`

`MelodicDiatonicInterval.direction_number`

`MelodicDiatonicInterval.direction_string`

`MelodicDiatonicInterval.harmonic_chromatic_interval`

`MelodicDiatonicInterval.harmonic_counterpoint_interval`

`MelodicDiatonicInterval.harmonic_diatonic_interval`

`MelodicDiatonicInterval.interval_class`

Inherited from `pitchtools.DiatonicIntervalObject`

`MelodicDiatonicInterval.interval_string`

Inherited from `pitchtools.DiatonicIntervalObject`

`MelodicDiatonicInterval.inversion_equivalent_chromatic_interval_class`

`MelodicDiatonicInterval.melodic_chromatic_interval`

`MelodicDiatonicInterval.melodic_counterpoint_interval`

`MelodicDiatonicInterval.melodic_diatonic_interval_class`

`MelodicDiatonicInterval.number`

Inherited from `pitchtools.DiatonicIntervalObject`

`MelodicDiatonicInterval.quality_string`

Inherited from `pitchtools.DiatonicIntervalObject`

`MelodicDiatonicInterval.semitones`

`MelodicDiatonicInterval.staff_spaces`

`MelodicDiatonicInterval.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`MelodicDiatonicInterval.__abs__()`

`MelodicDiatonicInterval.__add__(arg)`

`MelodicDiatonicInterval.__eq__(arg)`

Inherited from `pitchtools.DiatonicIntervalObject`

`MelodicDiatonicInterval.__float__()`

Inherited from `pitchtools.DiatonicIntervalObject`

`MelodicDiatonicInterval.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicDiatonicInterval.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MelodicDiatonicInterval.__hash__()`

Inherited from `pitchtools.IntervalObject`

`MelodicDiatonicInterval.__int__()`

Inherited from `pitchtools.DiatonicIntervalObject`

`MelodicDiatonicInterval.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MelodicDiatonicInterval.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

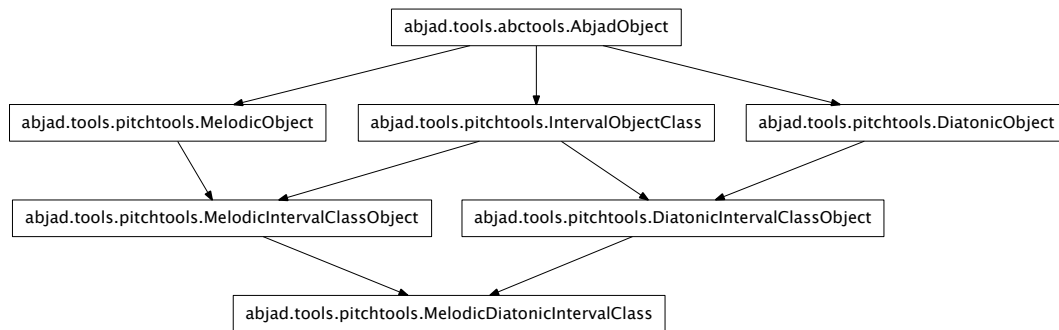
`MelodicDiatonicInterval.__mul__(arg)`

`MelodicDiatonicInterval.__ne__(arg)`

Inherited from `pitchtools.DiatonicIntervalObject`

```
MelodicDiatonicInterval.__neg__()
MelodicDiatonicInterval.__repr__()
MelodicDiatonicInterval.__rmul__(arg)
MelodicDiatonicInterval.__str__()
MelodicDiatonicInterval.__sub__(arg)
```

pitchtools.MelodicDiatonicIntervalClass



```
class pitchtools.MelodicDiatonicIntervalClass(*args)
    New in version 2.0. Abjad model of melodic diatonic interval-class:

    >>> pitchtools.MelodicDiatonicIntervalClass('-M9')
    MelodicDiatonicIntervalClass('-M2')
```

Melodic diatonic interval-classes are immutable.

Read-only Properties

`MelodicDiatonicIntervalClass.direction_number`

`MelodicDiatonicIntervalClass.direction_symbol`

`MelodicDiatonicIntervalClass.direction_word`

`MelodicDiatonicIntervalClass.number`

Inherited from `pitchtools.IntervalObjectClass`

`MelodicDiatonicIntervalClass.quality_string`

Inherited from `pitchtools.DiatonicIntervalClassObject`

`MelodicDiatonicIntervalClass.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`MelodicDiatonicIntervalClass.__abs__()`
Inherited from `pitchtools.DiatonicIntervalClassObject`

`MelodicDiatonicIntervalClass.__eq__(arg)`

`MelodicDiatonicIntervalClass.__float__()`
Inherited from `pitchtools.DiatonicIntervalClassObject`

`MelodicDiatonicIntervalClass.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`MelodicDiatonicIntervalClass.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`MelodicDiatonicIntervalClass.__hash__()`

`MelodicDiatonicIntervalClass.__int__()`
Inherited from `pitchtools.DiatonicIntervalClassObject`

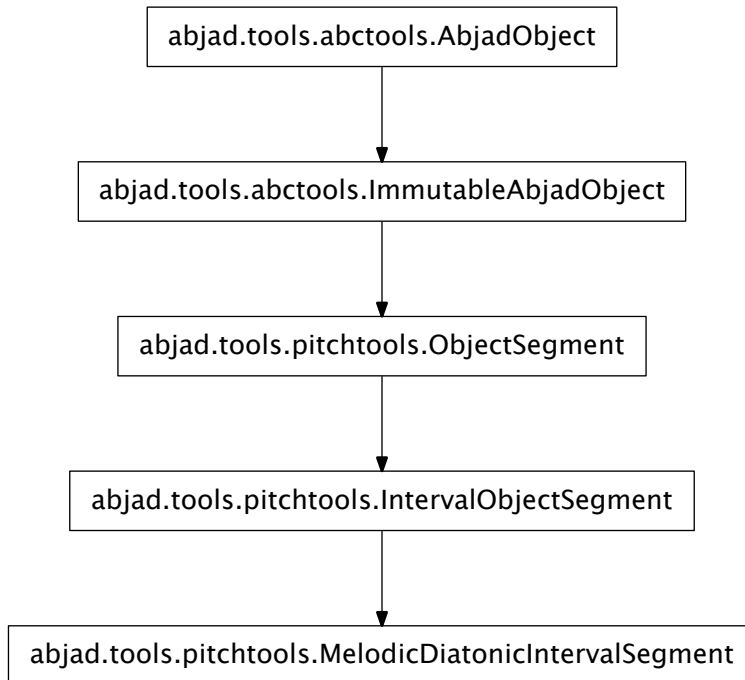
`MelodicDiatonicIntervalClass.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`MelodicDiatonicIntervalClass.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`MelodicDiatonicIntervalClass.__ne__(arg)`

`MelodicDiatonicIntervalClass.__repr__()`
Inherited from `pitchtools.DiatonicIntervalClassObject`

`MelodicDiatonicIntervalClass.__str__()`

pitchtools.MelodicDiatonicIntervalSegment

class `pitchtools.MelodicDiatonicIntervalSegment` (*args, **kwargs)

New in version 2.0. Abjad model of melodic diatonic interval segment:

```
>>> pitchtools.MelodicDiatonicIntervalSegment('M2 M9 -m3 -P4')
MelodicDiatonicIntervalSegment('+M2 +M9 -m3 -P4')
```

Melodic diatonic interval segments are immutable.

Read-only Properties

`MelodicDiatonicIntervalSegment.harmonic_chromatic_interval_segment`

`MelodicDiatonicIntervalSegment.harmonic_diatonic_interval_segment`

`MelodicDiatonicIntervalSegment.interval_classes`

Inherited from `pitchtools.IntervalObjectSegment`

`MelodicDiatonicIntervalSegment.intervals`

Inherited from `pitchtools.IntervalObjectSegment`

`MelodicDiatonicIntervalSegment.melodic_chromatic_interval_segment`

`MelodicDiatonicIntervalSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

MelodicDiatonicIntervalSegment.**count**(*value*) → integer – return number of occurrences of *value*

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of *value*.

Raises ValueError if the value is not present.

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**rotate**(*n*)

Inherited from `pitchtools.IntervalObjectSegment`

Special Methods

MelodicDiatonicIntervalSegment.**__add__**(*arg*)

Inherited from `pitchtools.ObjectSegment`

MelodicDiatonicIntervalSegment.**__contains__**()

x.**__contains__**(*y*) <==> *y* in *x*

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__eq__**()

x.**__eq__**(*y*) <==> *x*==*y*

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__ge__**()

x.**__ge__**(*y*) <==> *x*>=*y*

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__getitem__**()

x.**__getitem__**(*y*) <==> *x*[*y*]

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__getslice__**(*start*, *stop*)

Inherited from `pitchtools.ObjectSegment`

MelodicDiatonicIntervalSegment.**__gt__**()

x.**__gt__**(*y*) <==> *x*>*y*

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__hash__**() <==> *hash*(*x*)

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__iter__**() <==> *iter*(*x*)

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__le__**()

x.**__le__**(*y*) <==> *x*<=*y*

Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__len__**() <==> *len*(*x*)

Inherited from `__builtin__.tuple`

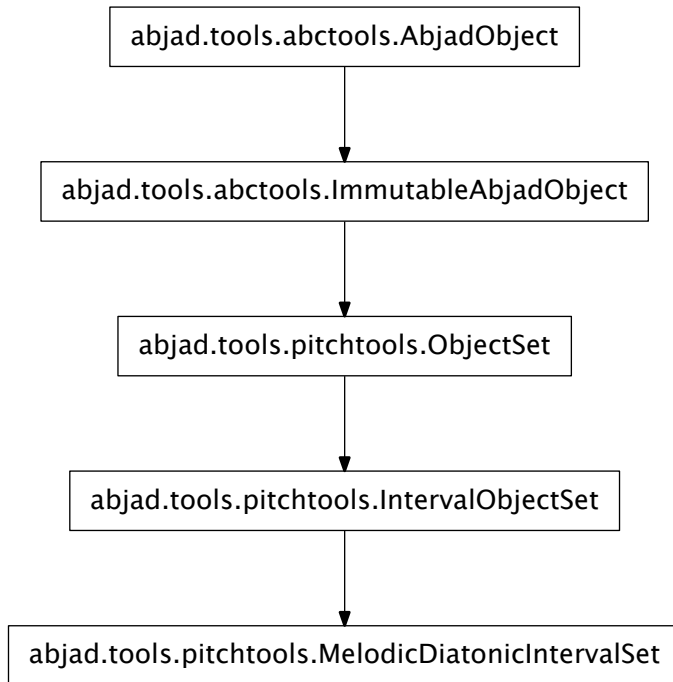
MelodicDiatonicIntervalSegment.**__lt__**()

x.**__lt__**(*y*) <==> *x*<*y*

Inherited from `__builtin__.tuple`

```
MelodicDiatonicIntervalSegment.__mul__(n)
    Inherited from pitchtools.ObjectSegment
MelodicDiatonicIntervalSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple
MelodicDiatonicIntervalSegment.__repr__()
MelodicDiatonicIntervalSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment
MelodicDiatonicIntervalSegment.__str__()
    Inherited from pitchtools.IntervalObjectSegment
```

pitchtools.MelodicDiatonicIntervalSet



```
class pitchtools.MelodicDiatonicIntervalSet(*args, **kwargs)
    New in version 2.0. Abjad model of melodic diatonic interval set:

    >>> pitchtools.MelodicDiatonicIntervalSet('M2 M2 -m3 -P4')
    MelodicDiatonicIntervalSet('-P4 -m3 +M2')
```

Melodic diatonic interval sets are immutable.

Read-only Properties

`MelodicDiatonicIntervalSet.harmonic_chromatic_interval_set`
`MelodicDiatonicIntervalSet.harmonic_diatonic_interval_set`
`MelodicDiatonicIntervalSet.melodic_chromatic_interval_set`
`MelodicDiatonicIntervalSet.melodic_diatonic_interval_numbers`
`MelodicDiatonicIntervalSet.melodic_diatonic_intervals`
`MelodicDiatonicIntervalSet.storage_format`
Storage format of Abjad object.
Return string.
Inherited from `abctools.AbjadObject`

Methods

`MelodicDiatonicIntervalSet.copy()`
Return a shallow copy of a set.
Inherited from `__builtin__.frozenset`
`MelodicDiatonicIntervalSet.difference()`
Return the difference of two or more sets as a new set.
(i.e. all elements that are in this set but not the others.)
Inherited from `__builtin__.frozenset`
`MelodicDiatonicIntervalSet.intersection()`
Return the intersection of two or more sets as a new set.
(i.e. elements that are common to all of the sets.)
Inherited from `__builtin__.frozenset`
`MelodicDiatonicIntervalSet.isdisjoint()`
Return True if two sets have a null intersection.
Inherited from `__builtin__.frozenset`
`MelodicDiatonicIntervalSet.issubset()`
Report whether another set contains this set.
Inherited from `__builtin__.frozenset`
`MelodicDiatonicIntervalSet.issuperset()`
Report whether this set contains another set.
Inherited from `__builtin__.frozenset`
`MelodicDiatonicIntervalSet.symmetric_difference()`
Return the symmetric difference of two sets as a new set.
(i.e. all elements that are in exactly one of the sets.)
Inherited from `__builtin__.frozenset`
`MelodicDiatonicIntervalSet.union()`
Return the union of sets as a new set.
(i.e. all elements that are in either set.)
Inherited from `__builtin__.frozenset`

Special Methods

```

MelodicDiatonicIntervalSet.__and__()
    x.__and__(y) <==> x&y
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__cmp__(y) <==> cmp(x, y)
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__contains__()
    x.__contains__(y) <==> y in x.
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__eq__()
    x.__eq__(y) <==> x==y
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__hash__() <==> hash(x)
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__iter__() <==> iter(x)
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset
MelodicDiatonicIntervalSet.__repr__()

```

```
MelodicDiatonicIntervalSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

MelodicDiatonicIntervalSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

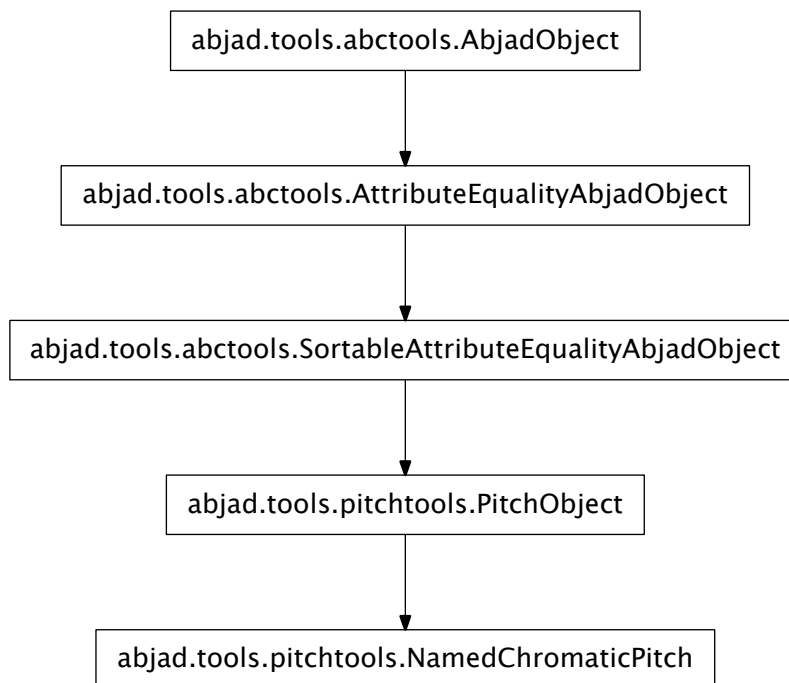
MelodicDiatonicIntervalSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

MelodicDiatonicIntervalSet.__str__()

MelodicDiatonicIntervalSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

MelodicDiatonicIntervalSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset
```

pitchtools.NamedChromaticPitch



class `pitchtools.NamedChromaticPitch(*args, **kwargs)`

New in version 1.1. Abjad model of named chromatic pitch:

```
>>> pitchtools.NamedChromaticPitch("cs' ")
NamedChromaticPitch("cs' ")
```

Named chromatic pitches are immutable.

Read-only Properties

`NamedChromaticPitch.accidental_spelling`

Read-only accidental spelling:

```
>>> pitchtools.NamedChromaticPitch("c").accidental_spelling
'mixed'
```

Return string.

`NamedChromaticPitch.chromatic_pitch_class_name`

Read-only chromatic pitch-class name:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.chromatic_pitch_class_name
'cs'
```

Return string.

`NamedChromaticPitch.chromatic_pitch_class_number`

Read-only chromatic pitch-class number:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.chromatic_pitch_class_number
1
```

Return integer or float.

`NamedChromaticPitch.chromatic_pitch_name`

Read-only chromatic pitch name:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.chromatic_pitch_name
"cs' "
```

Return string.

`NamedChromaticPitch.chromatic_pitch_number`

Read-only chromatic pitch-class number:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.chromatic_pitch_number
13
```

Return integer or float.

`NamedChromaticPitch.deviation_in_cents`

Read-only deviation of named chromatic pitch in cents:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.deviation_in_cents is None
True
```

Return integer or none.

`NamedChromaticPitch.diatonic_pitch_class_name`

Read-only diatonic pitch-class name:

```
>>> named_diatonic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_diatonic_pitch.diatonic_pitch_class_name
'c'
```

Return string.

`NamedChromaticPitch.diatonic_pitch_class_number`

Read-only diatonic pitch-class number:

```
>>> named_diatonic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_diatonic_pitch.diatonic_pitch_class_number
0
```

Return integer.

`NamedChromaticPitch.diatonic_pitch_name`

Read-only diatonic pitch name:

```
>>> named_diatonic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_diatonic_pitch.diatonic_pitch_name
"c' "
```

Return string.

`NamedChromaticPitch.diatonic_pitch_number`

Read-only diatonic pitch number:

```
>>> named_diatonic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_diatonic_pitch.diatonic_pitch_number
7
```

Return integer.

`NamedChromaticPitch.lilypond_format`

Read-only LilyPond input format of named chromatic pitch:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.lilypond_format
"cs' "
```

Return string.

`NamedChromaticPitch.named_chromatic_pitch_class`

Read-only named pitch-class:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.named_chromatic_pitch_class
NamedChromaticPitchClass('cs')
```

Return named chromatic pitch-class.

`NamedChromaticPitch.named_diatonic_pitch`

Read-only named diatonic pitch:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.named_diatonic_pitch
NamedDiatonicPitch("c' ")
```

Return named diatonic pitch.

NamedChromaticPitch.named_diatonic_pitch_class

Read-only named diatonic pitch-class:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

Return named diatonic pitch-class.

NamedChromaticPitch.numbered_chromatic_pitch

Read-only numbered chromatic pitch from named chromatic pitch:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.numbered_chromatic_pitch_class
NumberedChromaticPitchClass(1)
```

Return numbered chromatic pitch-class.

NamedChromaticPitch.numbered_chromatic_pitch_class

Read-only numbered pitch-class:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.numbered_chromatic_pitch_class
NumberedChromaticPitchClass(1)
```

Return numbered chromatic pitch-class.

NamedChromaticPitch.numbered_diatonic_pitch

Read-only numbered diatonic pitch:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.numbered_diatonic_pitch
NumberedDiatonicPitch(7)
```

Return numbered diatonic pitch.

NamedChromaticPitch.numbered_diatonic_pitch_class

Read-only numbered diatonic pitch-class:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

Return numbered diatonic pitch-class.

NamedChromaticPitch.octave_number

Read-only integer octave number:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.octave_number
5
```

Return integer.

NamedChromaticPitch.pitch_class_octave_label

Read-only pitch-class / octave label:

```
>>> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
>>> named_chromatic_pitch.pitch_class_octave_label
'C#5'
```

Return string.

`NamedChromaticPitch.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`NamedChromaticPitch.__abs__()`

`NamedChromaticPitch.__add__(melodic_interval)`

New in version 2.0.

`NamedChromaticPitch.__eq__(arg)`

`NamedChromaticPitch.__float__()`

`NamedChromaticPitch.__ge__(arg)`

`NamedChromaticPitch.__gt__(arg)`

`NamedChromaticPitch.__hash__()`

`NamedChromaticPitch.__int__()`

`NamedChromaticPitch.__le__(arg)`

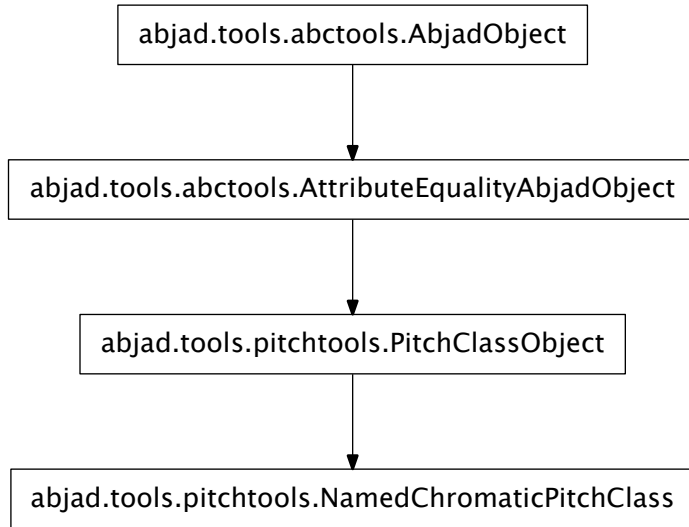
`NamedChromaticPitch.__lt__(arg)`

`NamedChromaticPitch.__ne__(arg)`

`NamedChromaticPitch.__repr__()`

`NamedChromaticPitch.__str__()`

`NamedChromaticPitch.__sub__(arg)`

pitchtools.NamedChromaticPitchClass

class `pitchtools.NamedChromaticPitchClass` (*arg*)

New in version 2.0. Abjad model of named chromatic pitch-class:

```
>>> npc = pitchtools.NamedChromaticPitchClass('cs')
```

```
>>> npc
NamedChromaticPitchClass('cs')
```

Named chromatic pitch-classes are immutable.

Read-only Properties

`NamedChromaticPitchClass.numbered_chromatic_pitch_class`

Read-only numbered chromatic pitch-class:

```
>>> npc.numbered_chromatic_pitch_class
NumberedChromaticPitchClass(1)
```

Return numbered chromatic pitch-class.

`NamedChromaticPitchClass.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NamedChromaticPitchClass.apply_accidental` (*accidental*)

Apply *accidental*:

```
>>> ncpc.apply_accidental('qs')
NamedChromaticPitchClass('ctqs')
```

Return named chromatic pitch-class.

NamedChromaticPitchClass.**transpose** (*melodic_diatonic_interval*)
 Transpose named chromatic pitch-class by *melodic_diatonic_interval*:

```
>>> ncpc.transpose(pitchtools.MelodicDiatonicInterval('major', 2))
NamedChromaticPitchClass('ds')
```

Return named chromatic pitch-class.

Special Methods

NamedChromaticPitchClass.**__abs__**()

NamedChromaticPitchClass.**__add__** (*melodic_diatonic_interval*)

NamedChromaticPitchClass.**__eq__** (*arg*)

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

NamedChromaticPitchClass.**__float__**()

NamedChromaticPitchClass.**__ge__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

NamedChromaticPitchClass.**__gt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

NamedChromaticPitchClass.**__hash__**()

Inherited from `pitchtools.PitchClassObject`

NamedChromaticPitchClass.**__int__**()

NamedChromaticPitchClass.**__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

NamedChromaticPitchClass.**__lt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

NamedChromaticPitchClass.**__ne__** (*arg*)

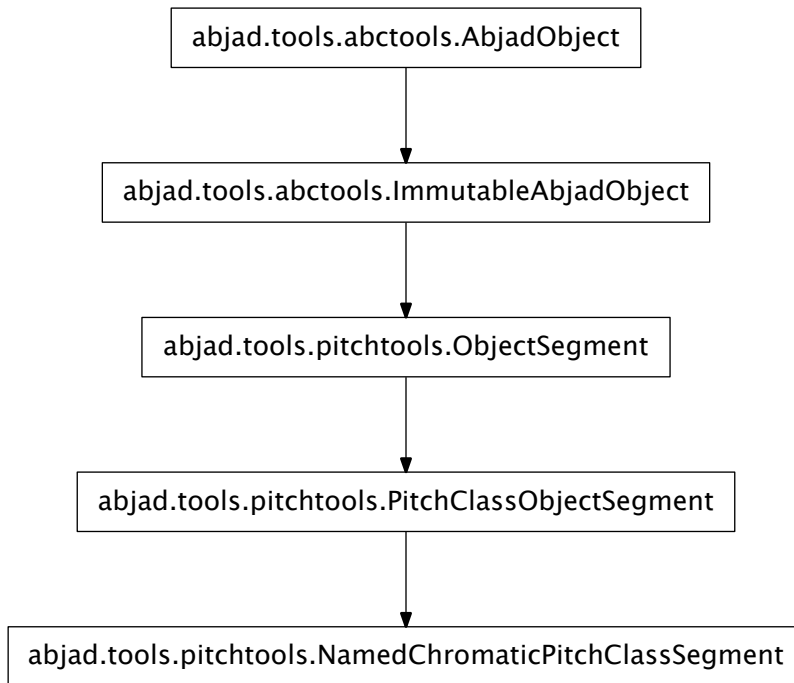
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`


```
NamedChromaticPitchClass.__repr__()
NamedChromaticPitchClass.__str__()
NamedChromaticPitchClass.__sub__(arg)
```

`pitchtools.NamedChromaticPitchClassSegment`



class `pitchtools.NamedChromaticPitchClassSegment` (*args, **kwargs)

New in version 2.0. Abjad model of named chromatic pitch-class segment:

```
>>> pitchtools.NamedChromaticPitchClassSegment(['gs', 'a', 'as', 'c', 'cs'])
NamedChromaticPitchClassSegment(['gs', 'a', 'as', 'c', 'cs'])
```

Named chromatic pitch-class segments are immutable.

Read-only Properties

```
NamedChromaticPitchClassSegment.inversion_equivalent_diatonic_interval_class_segment
NamedChromaticPitchClassSegment.named_chromatic_pitch_class_set
NamedChromaticPitchClassSegment.named_chromatic_pitch_classes
NamedChromaticPitchClassSegment.numbered_chromatic_pitch_class_segment
NamedChromaticPitchClassSegment.numbered_chromatic_pitch_class_set
NamedChromaticPitchClassSegment.numbered_chromatic_pitch_classes
```

`NamedChromaticPitchClassSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NamedChromaticPitchClassSegment.count(value)` \rightarrow integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.index(value[, start[, stop]])` \rightarrow integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.is_equivalent_under_transposition(arg)`

`NamedChromaticPitchClassSegment.retrograde()`

`NamedChromaticPitchClassSegment.rotate(n)`

`NamedChromaticPitchClassSegment.transpose(melodic_diatonic_interval)`

Special Methods

`NamedChromaticPitchClassSegment.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`NamedChromaticPitchClassSegment.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__getslice__(start, stop)`

Inherited from `pitchtools.ObjectSegment`

`NamedChromaticPitchClassSegment.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__hash__() <==> hash(x)`

Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__iter__() <==> iter(x)`

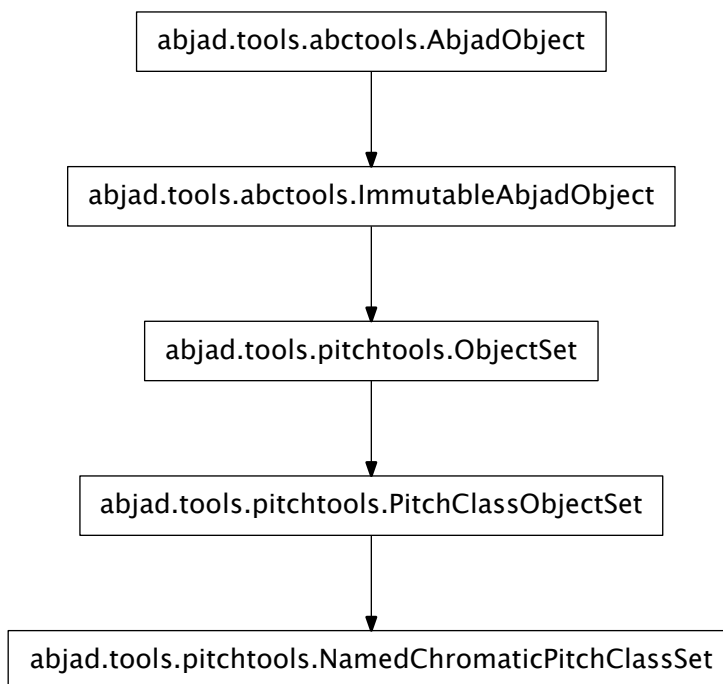
Inherited from `__builtin__.tuple`

```

NamedChromaticPitchClassSegment.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.tuple
NamedChromaticPitchClassSegment.__len__() <==> len(x)
    Inherited from __builtin__.tuple
NamedChromaticPitchClassSegment.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple
NamedChromaticPitchClassSegment.__mul__(n)
    Inherited from pitchtools.ObjectSegment
NamedChromaticPitchClassSegment.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple
NamedChromaticPitchClassSegment.__repr__()
NamedChromaticPitchClassSegment.__rmul__(n)
    Inherited from pitchtools.ObjectSegment
NamedChromaticPitchClassSegment.__str__()

```

`pitchtools.NamedChromaticPitchClassSet`



class `pitchtools.NamedChromaticPitchClassSet` (*args, **kwargs)

New in version 2.0. Abjad model of a named chromatic pitch-class set:

```
>>> named_chromatic_pitch_class_set = pitchtools.NamedChromaticPitchClassSet(
...     ['gs', 'g', 'as', 'c', 'cs'])

>>> named_chromatic_pitch_class_set
NamedChromaticPitchClassSet(['as', 'c', 'cs', 'g', 'gs'])

>>> print named_chromatic_pitch_class_set
{as, c, cs, g, gs}
```

Named chromatic pitch-class sets are immutable.

Read-only Properties

`NamedChromaticPitchClassSet.inversion_equivalent_diatonic_interval_class_vector`

`NamedChromaticPitchClassSet.named_chromatic_pitch_classes`

Read-only named chromatic pitch-classes:

```
>>> named_chromatic_pitch_class_set = pitchtools.NamedChromaticPitchClassSet(
...     ['gs', 'g', 'as', 'c', 'cs'])
>>> for x in named_chromatic_pitch_class_set.named_chromatic_pitch_classes: x
...
NamedChromaticPitchClass('as')
NamedChromaticPitchClass('c')
NamedChromaticPitchClass('cs')
NamedChromaticPitchClass('g')
NamedChromaticPitchClass('gs')
```

Return tuple.

`NamedChromaticPitchClassSet.numbered_chromatic_pitch_class_set`

`NamedChromaticPitchClassSet.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NamedChromaticPitchClassSet.copy()`

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.isdisjoint()`
 Return True if two sets have a null intersection.
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.issubset()`
 Report whether another set contains this set.
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.issuperset()`
 Report whether this set contains another set.
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.order_by(npc_seg)`

`NamedChromaticPitchClassSet.symmetric_difference()`
 Return the symmetric difference of two sets as a new set.
 (i.e. all elements that are in exactly one of the sets.)
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.transpose(melodic_diatonic_interval)`
 Transpose all npcs in self by melodic diatonic interval.

`NamedChromaticPitchClassSet.union()`
 Return the union of sets as a new set.
 (i.e. all elements that are in either set.)
 Inherited from `__builtin__.frozenset`

Special Methods

`NamedChromaticPitchClassSet.__and__()`
`x.__and__(y) <==> x&y`
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__cmp__(y) <==> cmp(x, y)`
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__contains__()`
`x.__contains__(y) <==> y in x.`
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__eq__(arg)`

`NamedChromaticPitchClassSet.__ge__()`
`x.__ge__(y) <==> x>=y`
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__gt__()`
`x.__gt__(y) <==> x>y`
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__hash__()`

`NamedChromaticPitchClassSet.__iter__()` <==> `iter(x)`
 Inherited from `__builtin__.frozenset`

```

NamedChromaticPitchClassSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset

NamedChromaticPitchClassSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset

NamedChromaticPitchClassSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

NamedChromaticPitchClassSet.__ne__(arg)

NamedChromaticPitchClassSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

NamedChromaticPitchClassSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

NamedChromaticPitchClassSet.__repr__()

NamedChromaticPitchClassSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

NamedChromaticPitchClassSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

NamedChromaticPitchClassSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

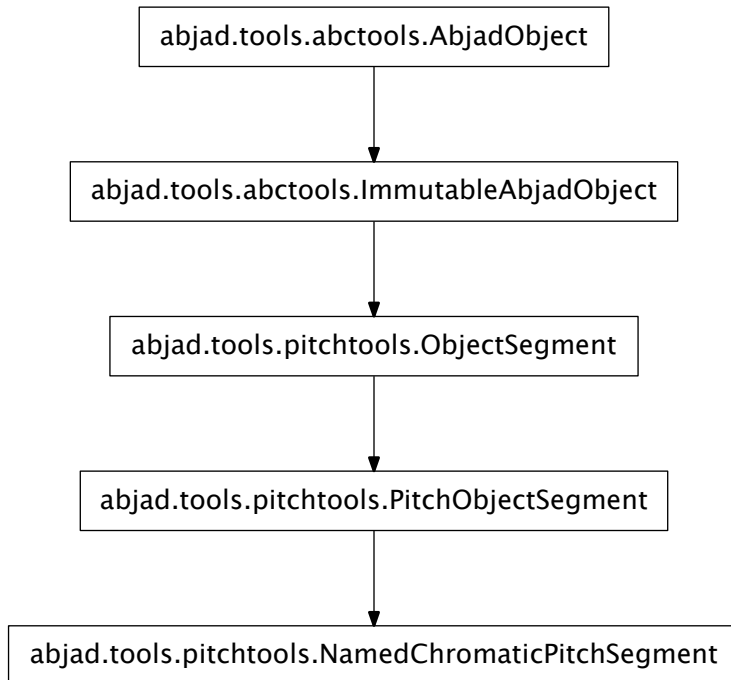
NamedChromaticPitchClassSet.__str__()

NamedChromaticPitchClassSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

NamedChromaticPitchClassSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset

```

pitchtools.NamedChromaticPitchSegment



class `pitchtools.NamedChromaticPitchSegment` (*args, **kwargs)

New in version 2.0. Abjad model of a named chromatic pitch segment:

```
>>> pitchtools.NamedChromaticPitchSegment(['bf', 'bqf', "fs'", "g'", 'bqf', "g'"])
NamedChromaticPitchSegment("bf bqf fs' g' bqf g'")
```

Named chromtic pitch segments are immutable.

Read-only Properties

`NamedChromaticPitchSegment.chromatic_pitch_numbers`

`NamedChromaticPitchSegment.harmonic_chromatic_interval_class_segment`

`NamedChromaticPitchSegment.harmonic_chromatic_interval_segment`

`NamedChromaticPitchSegment.harmonic_diatonic_interval_class_segment`

`NamedChromaticPitchSegment.harmonic_diatonic_interval_segment`

`NamedChromaticPitchSegment.inflection_point_count`

`NamedChromaticPitchSegment.inversion_equivalent_chromatic_interval_class_segment`

`NamedChromaticPitchSegment.inversion_equivalent_chromatic_interval_class_set`

`NamedChromaticPitchSegment.inversion_equivalent_chromatic_interval_class_vector`

`NamedChromaticPitchSegment.local_maxima`

`NamedChromaticPitchSegment.local_minima`
`NamedChromaticPitchSegment.melodic_chromatic_interval_class_segment`
`NamedChromaticPitchSegment.melodic_chromatic_interval_segment`
`NamedChromaticPitchSegment.melodic_diatonic_interval_class_segment`
`NamedChromaticPitchSegment.melodic_diatonic_interval_segment`
`NamedChromaticPitchSegment.named_chromatic_pitch_class_vector`
`NamedChromaticPitchSegment.named_chromatic_pitch_set`
`NamedChromaticPitchSegment.named_chromatic_pitch_vector`
`NamedChromaticPitchSegment.named_chromatic_pitches`
`NamedChromaticPitchSegment.numbered_chromatic_pitch_class_segment`
`NamedChromaticPitchSegment.numbered_chromatic_pitch_class_set`
`NamedChromaticPitchSegment.storage_format`
 Storage format of Abjad object.
 Return string.
 Inherited from `abctools.AbjadObject`

Methods

`NamedChromaticPitchSegment.count(value)` → integer – return number of occurrences of value
 Inherited from `__builtin__.tuple`
`NamedChromaticPitchSegment.index(value[, start[, stop]])` → integer – return first index of value.
 Raises `ValueError` if the value is not present.
 Inherited from `__builtin__.tuple`
`NamedChromaticPitchSegment.transpose(melodic_interval)`
 Transpose pitches in pitch segment by melodic interval and emit new pitch segment.

Special Methods

`NamedChromaticPitchSegment.__add__(arg)`
 Inherited from `pitchtools.ObjectSegment`
`NamedChromaticPitchSegment.__contains__()`
`x.__contains__(y) <==> y in x`
 Inherited from `__builtin__.tuple`
`NamedChromaticPitchSegment.__eq__()`
`x.__eq__(y) <==> x==y`
 Inherited from `__builtin__.tuple`
`NamedChromaticPitchSegment.__ge__()`
`x.__ge__(y) <==> x>=y`
 Inherited from `__builtin__.tuple`
`NamedChromaticPitchSegment.__getitem__()`
`x.__getitem__(y) <==> x[y]`
 Inherited from `__builtin__.tuple`

NamedChromaticPitchSegment.__**getslice**__(*start, stop*)

Inherited from pitchtools.ObjectSegment

NamedChromaticPitchSegment.__**gt**__()

x.__gt__(y) <==> x>y

Inherited from __builtin__.tuple

NamedChromaticPitchSegment.__**hash**__() <==> hash(x)

Inherited from __builtin__.tuple

NamedChromaticPitchSegment.__**iter**__() <==> iter(x)

Inherited from __builtin__.tuple

NamedChromaticPitchSegment.__**le**__()

x.__le__(y) <==> x<=y

Inherited from __builtin__.tuple

NamedChromaticPitchSegment.__**len**__() <==> len(x)

Inherited from __builtin__.tuple

NamedChromaticPitchSegment.__**lt**__()

x.__lt__(y) <==> x<y

Inherited from __builtin__.tuple

NamedChromaticPitchSegment.__**mul**__(*n*)

Inherited from pitchtools.ObjectSegment

NamedChromaticPitchSegment.__**ne**__()

x.__ne__(y) <==> x!=y

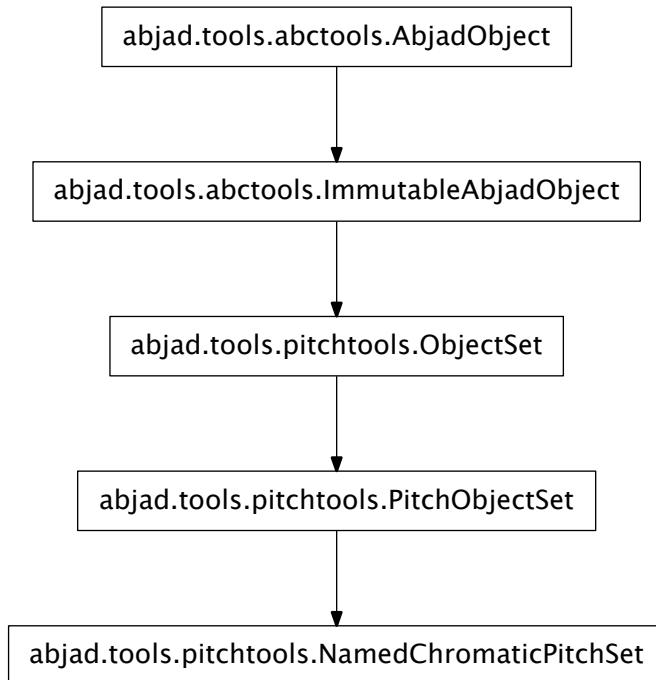
Inherited from __builtin__.tuple

NamedChromaticPitchSegment.__**repr**__()

NamedChromaticPitchSegment.__**rmul**__(*n*)

Inherited from pitchtools.ObjectSegment

pitchtools.NamedChromaticPitchSet



class `pitchtools.NamedChromaticPitchSet` (*args, **kwargs)

New in version 2.0. Abjad model of a named chromatic pitch set:

```
>>> pitchtools.NamedChromaticPitchSet(['bf', 'bqf', "fs'", "g'", 'bqf', "g'"])
NamedChromaticPitchSet(['bf', 'bqf', "fs'", "g'"])
```

Named chromatic pitch sets are immutable.

Read-only Properties

`NamedChromaticPitchSet.chromatic_pitch_numbers`

`NamedChromaticPitchSet.duplicate_pitch_classes`

`NamedChromaticPitchSet.is_pitch_class_unique`

`NamedChromaticPitchSet.named_chromatic_pitches`

`NamedChromaticPitchSet.numbered_chromatic_pitch_class_set`

`NamedChromaticPitchSet.numbered_chromatic_pitch_classes`

`NamedChromaticPitchSet.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NamedChromaticPitchSet.copy()`
 Return a shallow copy of a set.
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.difference()`
 Return the difference of two or more sets as a new set.
 (i.e. all elements that are in this set but not the others.)
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.intersection()`
 Return the intersection of two or more sets as a new set.
 (i.e. elements that are common to all of the sets.)
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.isdisjoint()`
 Return True if two sets have a null intersection.
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.issubset()`
 Report whether another set contains this set.
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.issuperset()`
 Report whether this set contains another set.
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.symmetric_difference()`
 Return the symmetric difference of two sets as a new set.
 (i.e. all elements that are in exactly one of the sets.)
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.transpose(n)`
 Transpose all pcs in self by n.

`NamedChromaticPitchSet.union()`
 Return the union of sets as a new set.
 (i.e. all elements that are in either set.)
 Inherited from `__builtin__.frozenset`

Special Methods

`NamedChromaticPitchSet.__and__()`
 $x._and_ (y) \iff x \& y$
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.__cmp__()`
 $x._cmp_ (y) \iff cmp(x, y)$
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.__contains__()`
 $x._contains_ (y) \iff y \text{ in } x$.
 Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.__eq__(arg)`

```
NamedChromaticPitchSet.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__hash__() <==> hash(x)
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__iter__() <==> iter(x)
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__len__() <==> len(x)
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__ne__(arg)

NamedChromaticPitchSet.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__repr__()

NamedChromaticPitchSet.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

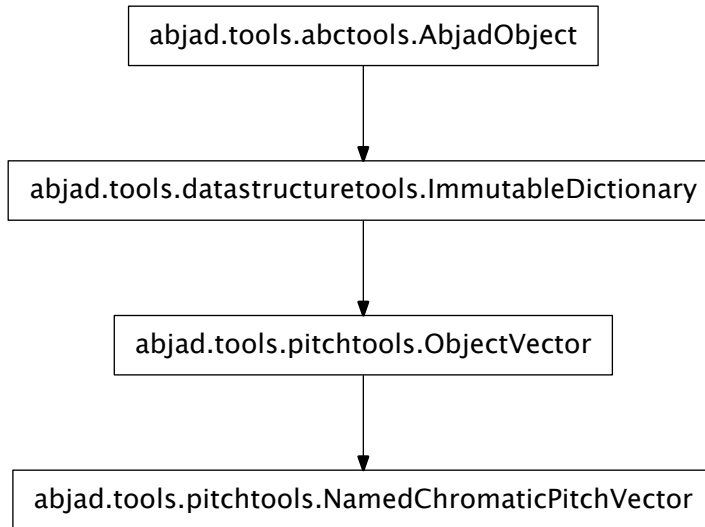
NamedChromaticPitchSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

NamedChromaticPitchSet.__str__()

NamedChromaticPitchSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset
```

```
NamedChromaticPitchSet.__xor__()
x.__xor__(y) <==> x^y
Inherited from __builtin__.frozenset
```

pitchtools.NamedChromaticPitchVector



class pitchtools.**NamedChromaticPitchVector** (*pitch_tokens*)

New in version 2.0. Abjad model of named chromatic pitch vector:

```
>>> named_chromatic_pitch_vector = pitchtools.NamedChromaticPitchVector(
...     ["c'", "c'", "cs'", "cs'", "cs'"])
```

```
>>> named_chromatic_pitch_vector
NamedChromaticPitchVector(c': 2, cs': 3)
```

```
>>> print named_chromatic_pitch_vector
NamedChromaticPitchVector(c': 2, cs': 3)
```

Named chromatic pitch vectors are immutable.

Read-only Properties

NamedChromaticPitchVector.**chromatic_pitch_numbers**

NamedChromaticPitchVector.**named_chromatic_pitches**

NamedChromaticPitchVector.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from abctools.AbjadObject

Methods

`NamedChromaticPitchVector.clear()` → None. Remove all items from D.

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.copy()` → a shallow copy of D

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.has_key(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.items()` → list of D's (key, value) pairs, as 2-tuples

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.iteritems()` → an iterator over the (key, value) items of D

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.iterkeys()` → an iterator over the keys of D

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.itervalues()` → an iterator over the values of D

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.keys()` → list of D's keys

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.update([E], **F)` → None. Update D from dict/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.values()` → list of D's values

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.viewitems()` → a set-like object providing a view on D's items

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.viewkeys()` → a set-like object providing a view on D's keys

Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.viewvalues()` → an object providing a view on D's values

Inherited from `__builtin__.dict`

Special Methods

`NamedChromaticPitchVector.__cmp__(y)` <==> `cmp(x, y)`

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__contains__(*k*) → True if D has a key *k*, else False

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__delitem__(**args*)

Inherited from `datastructuretools.ImmutableDictionary`

NamedChromaticPitchVector.__eq__()

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__ge__()

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__getitem__()

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__gt__()

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__iter__() <==> *iter(x)*

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__le__()

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__len__() <==> *len(x)*

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__lt__()

`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__ne__()

`x.__ne__(y) <==> x!=y`

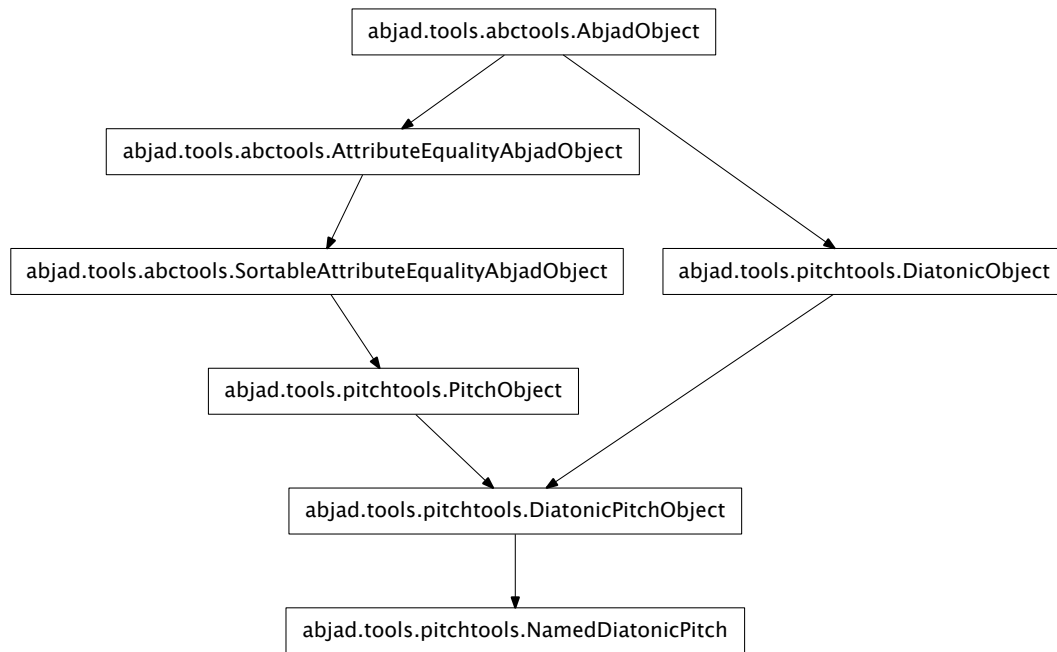
Inherited from `__builtin__.dict`

NamedChromaticPitchVector.__repr__()

NamedChromaticPitchVector.__setitem__(**args*)

Inherited from `datastructuretools.ImmutableDictionary`

pitchtools.NamedDiatonicPitch



class `pitchtools.NamedDiatonicPitch` (*arg*)

New in version 2.0. Abjad model of a named diatonic pitch:

```
>>> named_diatonic_pitch = pitchtools.NamedDiatonicPitch("c' ' ")
```

```
>>> named_diatonic_pitch
NamedDiatonicPitch("c' ' ")
```

```
>>> print named_diatonic_pitch
c' '
```

Named diatonic pitches are immutable.

Read-only Properties

`NamedDiatonicPitch.chromatic_pitch_class_name`

Read-only chromatic pitch-class name:

```
>>> pitchtools.NamedDiatonicPitch("c' ' ").chromatic_pitch_class_name
'c'
```

Return string.

`NamedDiatonicPitch.chromatic_pitch_class_number`

Read-only chromatic pitch-class number:

```
>>> pitchtools.NamedDiatonicPitch("c' ' ").chromatic_pitch_class_number
0
```


Return integer.

`NamedDiatonicPitch.chromatic_pitch_name`

Read-only chromatic pitch name:

```
>>> pitchtools.NamedDiatonicPitch("c'").chromatic_pitch_name
"c' "
```

Return string.

`NamedDiatonicPitch.chromatic_pitch_number`

Read-only chromatic pitch number:

```
>>> pitchtools.NamedDiatonicPitch("c'").chromatic_pitch_number
12
```

Return integer.

`NamedDiatonicPitch.diatonic_pitch_class_name`

Read-only diatonic pitch-class name:

```
>>> pitchtools.NamedDiatonicPitch("c'").diatonic_pitch_class_name
'c'
```

Return string.

`NamedDiatonicPitch.diatonic_pitch_class_number`

Read-only diatonic pitch-class number:

```
>>> pitchtools.NamedDiatonicPitch("c'").diatonic_pitch_class_number
0
```

Return integer.

`NamedDiatonicPitch.diatonic_pitch_name`

Read-only diatonic pitch name:

```
>>> pitchtools.NamedDiatonicPitch("c'").diatonic_pitch_name
"c' "
```

Return string.

`NamedDiatonicPitch.diatonic_pitch_number`

Read-only diatonic pitch number:

```
>>> pitchtools.NamedDiatonicPitch("c'").diatonic_pitch_number
7
```

Return integer.

`NamedDiatonicPitch.lilypond_format`

Read-only LilyPond input format of named diatonic pitch:

```
>>> pitchtools.NamedDiatonicPitch("c'").lilypond_format
"c' "
```

Return string.

`NamedDiatonicPitch.named_chromatic_pitch`

Read-only named chromatic pitch:

```
>>> pitchtools.NamedDiatonicPitch("c'").named_chromatic_pitch
NamedChromaticPitch("c' ")
```

Return named chromatic pitch.

`NamedDiatonicPitch.named_chromatic_pitch_class`

Read-only named chromatic pitch-class:

```
>>> pitchtools.NamedDiatonicPitch("c'").named_chromatic_pitch_class
NamedChromaticPitchClass('c')
```

Return named chromatic pitch-class.

`NamedDiatonicPitch.named_diatonic_pitch_class`

Read-only named diatonic pitch-class:

```
>>> pitchtools.NamedDiatonicPitch("c'").named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

Return named diatonic pitch-class.

`NamedDiatonicPitch.numbered_chromatic_pitch`

Read-only numbered chromatic pitch:

```
>>> pitchtools.NamedDiatonicPitch("c'").numbered_chromatic_pitch
NumberedChromaticPitch(12)
```

Return numbered chromatic pitch.

`NamedDiatonicPitch.numbered_chromatic_pitch_class`

Read-only numbered chromatic pitch-class:

```
>>> pitchtools.NamedDiatonicPitch("c'").numbered_chromatic_pitch_class
NumberedChromaticPitchClass(0)
```

Return numbered chromatic pitch-class.

`NamedDiatonicPitch.numbered_diatonic_pitch`

Read-only numbered diatonic pitch:

```
>>> pitchtools.NamedDiatonicPitch("c'").numbered_diatonic_pitch
NumberedDiatonicPitch(7)
```

Return numbered diatonic pitch.

`NamedDiatonicPitch.numbered_diatonic_pitch_class`

Read-only numbered diatonic pitch-class:

```
>>> pitchtools.NamedDiatonicPitch("c'").numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

Return numbered diatonic pitch-class.

`NamedDiatonicPitch.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`NamedDiatonicPitch.__abs__()`

`NamedDiatonicPitch.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`NamedDiatonicPitch.__float__()`

`NamedDiatonicPitch.__ge__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NamedDiatonicPitch.__gt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NamedDiatonicPitch.__hash__()`

Inherited from `pitchtools.PitchObject`

`NamedDiatonicPitch.__int__()`

`NamedDiatonicPitch.__le__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NamedDiatonicPitch.__lt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

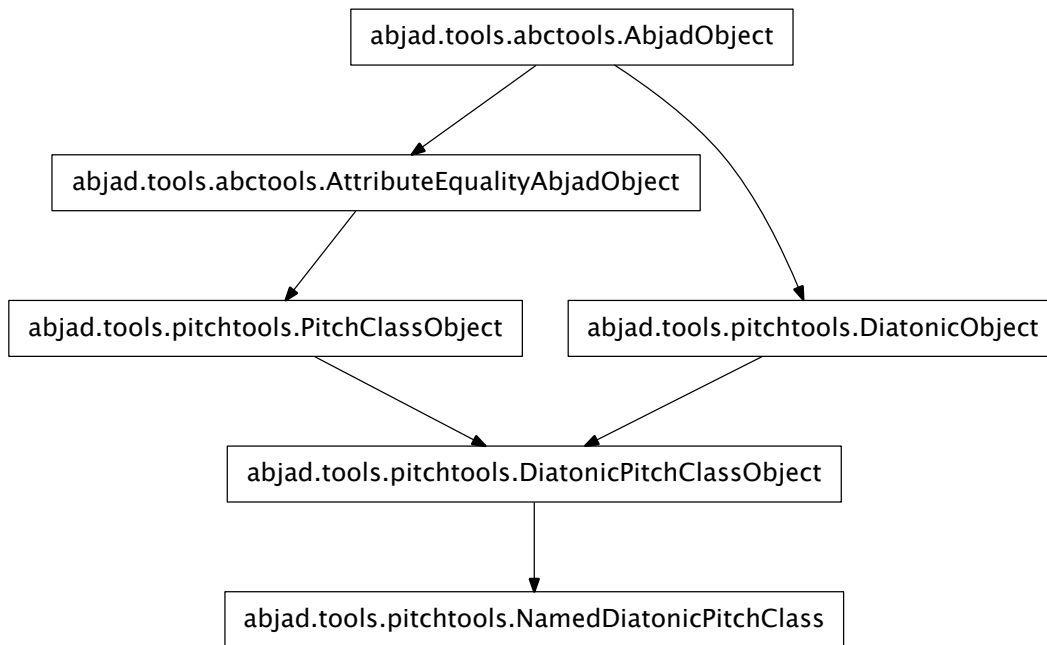
`NamedDiatonicPitch.__ne__(arg)`

Inherited from `pitchtools.PitchObject`

`NamedDiatonicPitch.__repr__()`

`NamedDiatonicPitch.__str__()`

pitchtools.NamedDiatonicPitchClass



class `pitchtools.NamedDiatonicPitchClass` (*arg*)
 New in version 2.0. Abjad model of a named diatonic pitch-class:

```
>>> pitchtools.NamedDiatonicPitchClass('c')
NamedDiatonicPitchClass('c')
```

Named diatonic pitch-classes are immutable.

Read-only Properties

`NamedDiatonicPitchClass.numbered_diatonic_pitch_class`
 Read-only numbered diatonic pitch-class from named diatonic pitch-class:

```
>>> named_diatonic_pitch_class = pitchtools.NamedDiatonicPitchClass('c')
>>> named_diatonic_pitch_class.numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

Return numbered diatonic pitch-class.

`NamedDiatonicPitchClass.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`NamedDiatonicPitchClass.__abs__()`

`NamedDiatonicPitchClass.__eq__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.
Inherited from `abctools.AttributeEqualityAbjadObject`

`NamedDiatonicPitchClass.__float__()`

`NamedDiatonicPitchClass.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`NamedDiatonicPitchClass.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`NamedDiatonicPitchClass.__hash__()`
Inherited from `pitchtools.PitchClassObject`

`NamedDiatonicPitchClass.__int__()`

`NamedDiatonicPitchClass.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

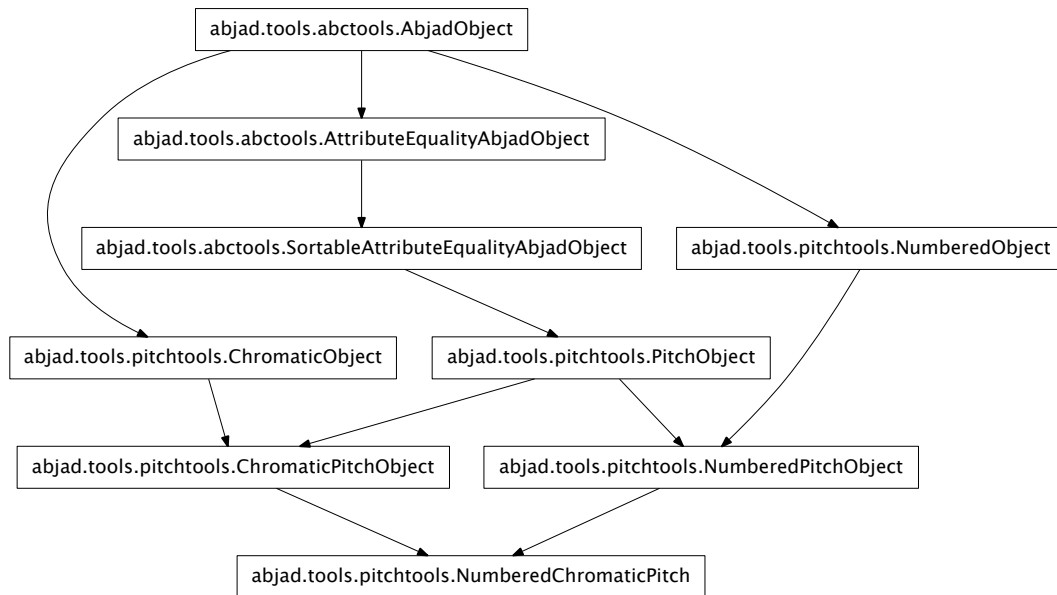
`NamedDiatonicPitchClass.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`NamedDiatonicPitchClass.__ne__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.
Inherited from `abctools.AttributeEqualityAbjadObject`

`NamedDiatonicPitchClass.__repr__()`

`NamedDiatonicPitchClass.__str__()`

pitchtools.NumberedChromaticPitch



class `pitchtools.NumberedChromaticPitch` (*arg*)
 New in version 2.0. Abjad model of a numbered chromatic pitch:

```
>>> pitchtools.NumberedChromaticPitch(13)
NumberedChromaticPitch(13)
```

Numbered chromatic pitches are immutable.

Read-only Properties

`NumberedChromaticPitch.chromatic_pitch_number`

Read-only chromatic pitch-class number:

```
>>> pitchtools.NumberedChromaticPitch(13).chromatic_pitch_number
13
```

Return integer or float.

`NumberedChromaticPitch.diatonic_pitch_class_number`

Read-only diatonic pitch-class number:

```
>>> pitchtools.NumberedChromaticPitch(13).diatonic_pitch_class_number
0
```

Return integer.

`NumberedChromaticPitch.diatonic_pitch_number`

Read-only diatonic pitch-class number:

```
>>> pitchtools.NumberedChromaticPitch(13).diatonic_pitch_number
7
```

Return integer.

`NumberedChromaticPitch.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NumberedChromaticPitch.apply_accidental (accidental=None)`

Apply *accidental*:

```
>>> pitchtools.NumberedChromaticPitch(13).apply_accidental('flat')
NumberedChromaticPitch(12)
```

Return numbered chromatic pitch.

`NumberedChromaticPitch.transpose (n=0)`

Transpose by *n* semitones:

```
>>> pitchtools.NumberedChromaticPitch(13).transpose(1)
NumberedChromaticPitch(14)
```

Return numbered chromatic pitch.

Special Methods

`NumberedChromaticPitch.__abs__()`

`NumberedChromaticPitch.__add__(arg)`

`NumberedChromaticPitch.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedChromaticPitch.__float__()`

`NumberedChromaticPitch.__ge__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedChromaticPitch.__gt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedChromaticPitch.__hash__()`

`NumberedChromaticPitch.__int__()`

`NumberedChromaticPitch.__le__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedChromaticPitch.__lt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedChromaticPitch.__ne__(arg)`

Inherited from `pitchtools.PitchObject`

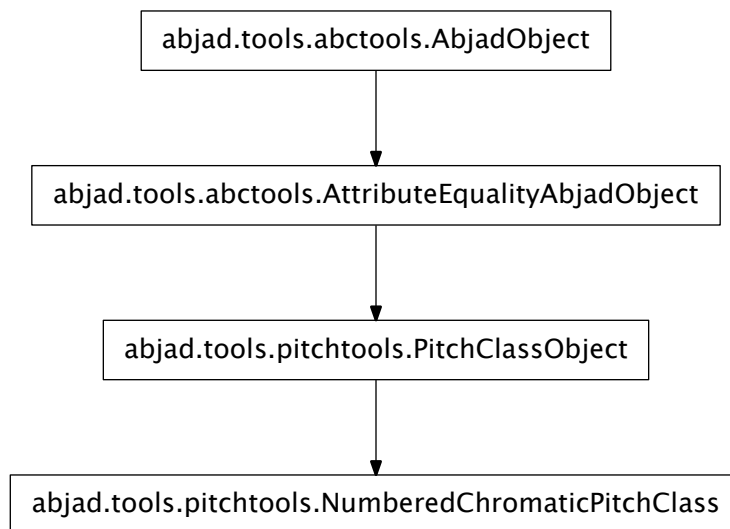
`NumberedChromaticPitch.__neg__()`

`NumberedChromaticPitch.__repr__()`

`NumberedChromaticPitch.__str__()`

`NumberedChromaticPitch.__sub__(arg)`

`pitchtools.NumberedChromaticPitchClass`



class `pitchtools.NumberedChromaticPitchClass(arg)`

New in version 2.0. Abjad model of a numbered chromatic pitch-class:

```
>>> pitchtools.NumberedChromaticPitchClass(13)
NumberedChromaticPitchClass(1)
```

Numbered chromatic pitch-classes are immutable.

Read-only Properties

`NumberedChromaticPitchClass.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NumberedChromaticPitchClass.apply_accidental (accidental=None)`

Emit new numbered chromatic pitch-class as sum of self and accidental.

`NumberedChromaticPitchClass.invert ()`

Invert pitch-class.

`NumberedChromaticPitchClass.multiply (n)`

Multiply pitch-class by n.

`NumberedChromaticPitchClass.transpose (n)`

Transpose pitch-class by n.

Special Methods

`NumberedChromaticPitchClass.__abs__ ()`

`NumberedChromaticPitchClass.__add__ (arg)`

Addition defined against melodic chromatic intervals only.

`NumberedChromaticPitchClass.__eq__ (arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedChromaticPitchClass.__float__ ()`

`NumberedChromaticPitchClass.__ge__ (arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClass.__gt__ (arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClass.__hash__ ()`

Inherited from `pitchtools.PitchClassObject`

`NumberedChromaticPitchClass.__int__ ()`

`NumberedChromaticPitchClass.__le__ (arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClass.__lt__ (arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClass.__ne__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedChromaticPitchClass.__neg__()`

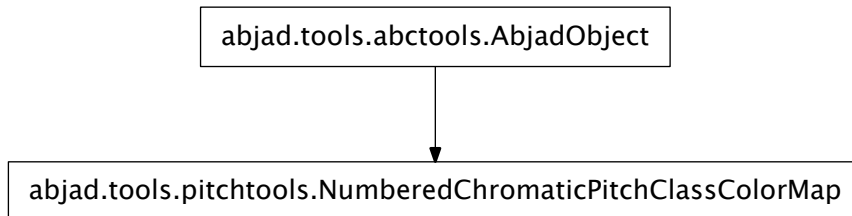
`NumberedChromaticPitchClass.__repr__()`

`NumberedChromaticPitchClass.__str__()`

`NumberedChromaticPitchClass.__sub__(arg)`

Subtraction defined against both melodic chromatic intervals and against other pitch-classes.

`pitchtools.NumberedChromaticPitchClassColorMap`



class `pitchtools.NumberedChromaticPitchClassColorMap` (*pitch_iterables, colors*)

New in version 2.0. Abjad model of a numbered chromatic pitch-class color map:

```

>>> chromatic_pitch_class_numbers = [[-8, 2, 10, 21], [0, 11, 32, 41], [15, 25, 42, 43]]
>>> colors = ['red', 'green', 'blue']
>>> mapping = pitchtools.NumberedChromaticPitchClassColorMap(
... chromatic_pitch_class_numbers, colors)
  
```

Numbered chromatic pitch-class color maps are immutable.

Read-only Properties

`NumberedChromaticPitchClassColorMap.colors`

`NumberedChromaticPitchClassColorMap.pairs`

`NumberedChromaticPitchClassColorMap.pitch_iterables`

`NumberedChromaticPitchClassColorMap.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClassColorMap.twelve_tone_complete`

`NumberedChromaticPitchClassColorMap.twenty_four_tone_complete`

Methods

`NumberedChromaticPitchClassColorMap.get(key, alternative=None)`

Special Methods

`NumberedChromaticPitchClassColorMap.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClassColorMap.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClassColorMap.__getitem__(pc)`

`NumberedChromaticPitchClassColorMap.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClassColorMap.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClassColorMap.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClassColorMap.__ne__(arg)`

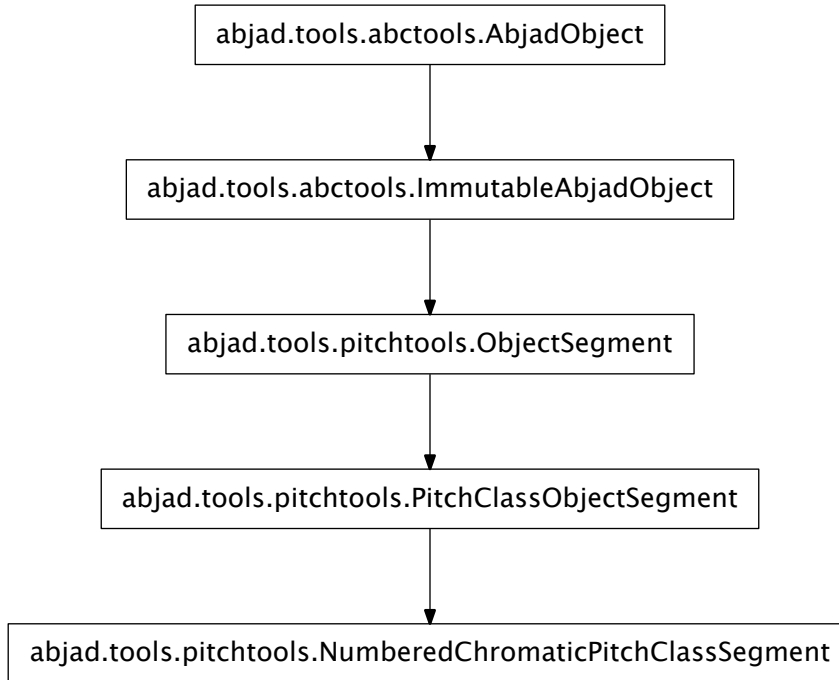
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClassColorMap.__repr__()`

pitchtools.NumberedChromaticPitchClassSegment



class `pitchtools.NumberedChromaticPitchClassSegment` (*args, **kwargs)

New in version 2.0. Abjad model of a numbered chromatic pitch-class segment:

```
>>> pitchtools.NumberedChromaticPitchClassSegment([-2, -1.5, 6, 7, -1.5, 7])
NumberedChromaticPitchClassSegment([10, 10.5, 6, 7, 10.5, 7])
```

Numbered chromatic pitch-class segments are immutable.

Read-only Properties

`NumberedChromaticPitchClassSegment.inversion_equivalent_chromatic_interval_class_segment`

Read-only inversion-equivalent chromatic interval-class segment:

```
>>> segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5, 6, 7, 10.5, 7])

>>> segment.inversion_equivalent_chromatic_interval_class_segment
InversionEquivalentChromaticIntervalClassSegment(0.5, 4.5, 1, 3.5, 3.5)
```

Return inversion-equivalent chromatic interval-class segment.

`NumberedChromaticPitchClassSegment.numbered_chromatic_pitch_class_set`

Read-only numbered chromatic pitch-class set from numbered chromatic pitch-class segment:

```
>>> segment.numbered_chromatic_pitch_class_set
NumberedChromaticPitchClassSet([6, 7, 10, 10.5])
```

Return numbered chromatic pitch-class set.

`NumberedChromaticPitchClassSegment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NumberedChromaticPitchClassSegment.alpha()`

Morris alpha transform of numbered chromatic pitch-class segment:

```
>>> segment.alpha()
NumberedChromaticPitchClassSegment([11, 11.5, 7, 6, 11.5, 6])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.invert()`

Invert numbered chromatic pitch-class segment:

```
>>> segment.invert()
NumberedChromaticPitchClassSegment([2, 1.5, 6, 5, 1.5, 5])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment.multiply(n)`

Multiply numbered chromatic pitch-class segment by *n*:

```
>>> segment.multiply(5)
NumberedChromaticPitchClassSegment([2, 4.5, 6, 11, 4.5, 11])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment.retrograde()`

Retrograde of numbered chromatic pitch-class segment:

```
>>> segment.retrograde()
NumberedChromaticPitchClassSegment([7, 10.5, 7, 6, 10.5, 10])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment.rotate(n)`

Rotate numbered chromatic pitch-class segment:

```
>>> segment.rotate(1)
NumberedChromaticPitchClassSegment([7, 10, 10.5, 6, 7, 10.5])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment.transpose(n)`

Transpose numbered chromatic pitch-class segment:

```
>>> segment.transpose(10)
NumberedChromaticPitchClassSegment([8, 8.5, 4, 5, 8.5, 5])
```

Return numbered chromatic pitch-class segment.

Special Methods

`NumberedChromaticPitchClassSegment.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`NumberedChromaticPitchClassSegment.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__getslice__(start, stop)`

Inherited from `pitchtools.ObjectSegment`

`NumberedChromaticPitchClassSegment.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__hash__() <==> hash(x)`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__iter__() <==> iter(x)`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__le__()`

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__len__() <==> len(x)`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__lt__()`

`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__mul__(n)`

Inherited from `pitchtools.ObjectSegment`

`NumberedChromaticPitchClassSegment.__ne__()`

`x.__ne__(y) <==> x!=y`

Inherited from `__builtin__.tuple`

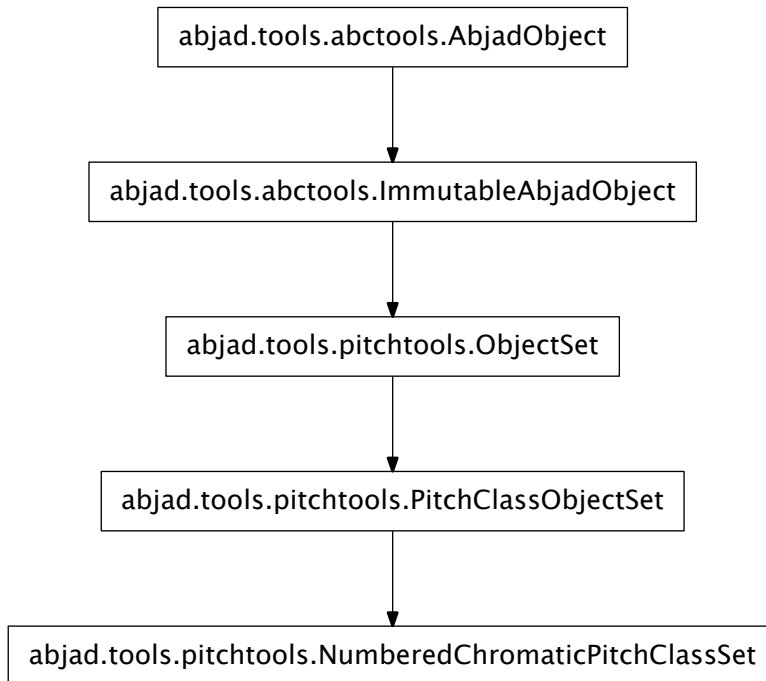
`NumberedChromaticPitchClassSegment.__repr__()`

`NumberedChromaticPitchClassSegment.__rmul__(n)`

Inherited from `pitchtools.ObjectSegment`

NumberedChromaticPitchClassSegment.__str__()

pitchtools.NumberedChromaticPitchClassSet



class pitchtools.**NumberedChromaticPitchClassSet** (*args, **kwargs)

New in version 2.0. Abjad model of a numbered chromatic pitch-class set:

```
>>> ncpcs = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5, 6, 7, -1.5, 7])
```

```
>>> ncpcs
NumberedChromaticPitchClassSet([6, 7, 10, 10.5])
```

```
>>> print ncpcs
{6, 7, 10, 10.5}
```

Numbered chromatic pitch-class sets are immutable.

Read-only Properties

NumberedChromaticPitchClassSet.**inversion_equivalent_chromatic_interval_class_set**

Read-only inversion-equivalent chromatic interval-class set:

```
>>> ncpcs.inversion_equivalent_chromatic_interval_class_set
InversionEquivalentChromaticIntervalClassSet(0.5, 1, 3, 3.5, 4, 4.5)
```

Return inversion-equivalent chromatic interval-class set.

`NumberedChromaticPitchClassSet.inversion_equivalent_chromatic_interval_class_vector`

Read-only inversion-equivalent chromatic interval-class vector:

```
>>> ncpcs.inversion_equivalent_chromatic_interval_class_vector
InversionEquivalentChromaticIntervalClassVector(0 | 1 0 1 1 0 0 1 0 0 1 1 0)
```

Return inversion-equivalent chromatic interval-class vector.

`NumberedChromaticPitchClassSet.numbered_chromatic_pitch_classes`

Read-only numbered chromatic pitch-classes:

```
>>> result = ncpcs.numbered_chromatic_pitch_classes

>>> for x in result: x
...
NumberedChromaticPitchClass(6)
NumberedChromaticPitchClass(7)
NumberedChromaticPitchClass(10)
NumberedChromaticPitchClass(10.5)
```

Return tuple.

`NumberedChromaticPitchClassSet.prime_form`

To be implemented.

`NumberedChromaticPitchClassSet.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NumberedChromaticPitchClassSet.copy()`

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.invert()`

Invert numbered chromatic pitch-class set:

```
>>> ncpcs.invert()
NumberedChromaticPitchClassSet([1.5, 2, 5, 6])
```

Return numbered chromatic pitch-class set.

`NumberedChromaticPitchClassSet.is_transposed_subset(pcset)`

True when self is transposed subset of *pcset*. False otherwise:


```
>>> pcset_1 = pitchtools.NumberedChromaticPitchClassSet (
... [-2, -1.5, 6, 7, -1.5, 7])
>>> pcset_2 = pitchtools.NumberedChromaticPitchClassSet (
... [-2, -1.5, 6, 7, -1.5, 7, 7.5, 8])

>>> pcset_1.is_transposed_subset(pcset_2)
True
```

Return boolean.

`NumberedChromaticPitchClassSet.is_transposed_superset(pcset)`
 True when self is transposed superset of *pcset*. False otherwise:

```
>>> pcset_1 = pitchtools.NumberedChromaticPitchClassSet (
... [-2, -1.5, 6, 7, -1.5, 7])
>>> pcset_2 = pitchtools.NumberedChromaticPitchClassSet (
... [-2, -1.5, 6, 7, -1.5, 7, 7.5, 8])

>>> pcset_2.is_transposed_superset(pcset_1)
True
```

Return boolean.

`NumberedChromaticPitchClassSet.isdisjoint()`
 Return True if two sets have a null intersection.

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.issubset()`
 Report whether another set contains this set.

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.issuperset()`
 Report whether this set contains another set.

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.multiply(n)`
 Multiply numbered chromatic pitch-class set by *n*:

```
>>> ncpcs.multiply(5)
NumberedChromaticPitchClassSet([2, 4.5, 6, 11])
```

Return numbered chromatic pitch-class set.

`NumberedChromaticPitchClassSet.symmetric_difference()`
 Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.transpose(n)`
 Transpose numbered chromatic pitch-class set by *n*:

```
>>> ncpcs.transpose(5)
NumberedChromaticPitchClassSet([0, 3, 3.5, 11])
```

Return numbered chromatic pitch-class set.

`NumberedChromaticPitchClassSet.union()`
 Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Inherited from `__builtin__.frozenset`

Special Methods

`NumberedChromaticPitchClassSet.__and__()`

`x.__and__(y) <==> x&y`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__cmp__(y) <==> cmp(x, y)`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__contains__()`

`x.__contains__(y) <==> y in x.`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__eq__(arg)`

`NumberedChromaticPitchClassSet.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__hash__()`

`NumberedChromaticPitchClassSet.__iter__() <==> iter(x)`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__le__()`

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__len__() <==> len(x)`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__lt__()`

`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__ne__(arg)`

`NumberedChromaticPitchClassSet.__or__()`

`x.__or__(y) <==> x|y`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__rand__()`

`x.__rand__(y) <==> y&x`

Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__repr__()`

`NumberedChromaticPitchClassSet.__ror__()`

`x.__ror__(y) <==> y|x`

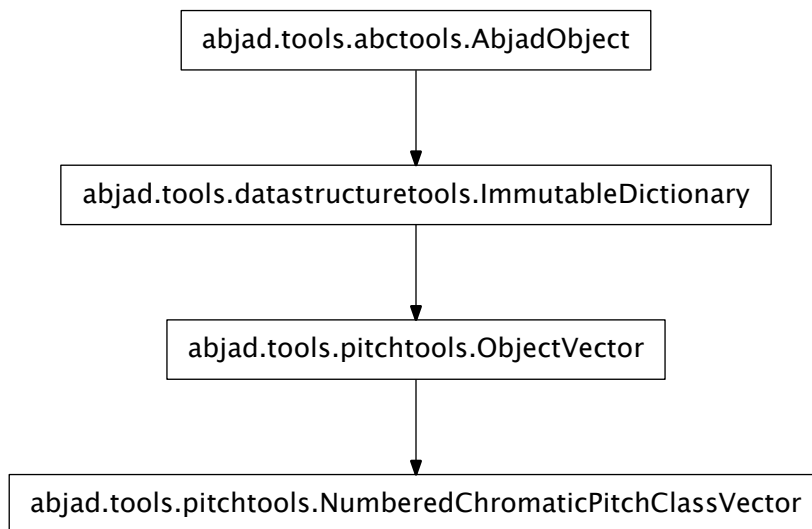
Inherited from `__builtin__.frozenset`

```

NumberedChromaticPitchClassSet.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset
NumberedChromaticPitchClassSet.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset
NumberedChromaticPitchClassSet.__str__()
NumberedChromaticPitchClassSet.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset
NumberedChromaticPitchClassSet.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset

```

pitchtools.NumberedChromaticPitchClassVector



class pitchtools.**NumberedChromaticPitchClassVector** (*pitch_class_tokens*)

New in version 2.0. Abjad model of numbered chromatic pitch-class vector:

```

>>> ncpcv = pitchtools.NumberedChromaticPitchClassVector(
...     [13, 13, 14.5, 14.5, 14.5, 6, 6, 6])

>>> print ncpcv
0 2 0 0 0 0 | 3 0 0 0 0 0
0 0 3 0 0 0 | 0 0 0 0 0 0

```

Numbered chromatic pitch-class vectors are immutable.

Read-only Properties

`NumberedChromaticPitchClassVector.chromatic_pitch_class_numbers`

Read-only chromatic pitch-class numbers from numbered chromatic pitch-class vector:

```
>>> ncpvc.chromatic_pitch_class_numbers
[1, 2.5, 6]
```

Return list.

`NumberedChromaticPitchClassVector.numbered_chromatic_pitch_classes`

Read-only numbered chromatic pitch-classes from numbered chromatic pitch-class vector:

```
>>> result = ncpvc.numbered_chromatic_pitch_classes

>>> for x in result: x
...
NumberedChromaticPitchClass(2.5)
NumberedChromaticPitchClass(1)
NumberedChromaticPitchClass(6)
```

Return list.

`NumberedChromaticPitchClassVector.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NumberedChromaticPitchClassVector.clear()` → None. Remove all items from D.

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.copy()` → a shallow copy of D

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.has_key(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.items()` → list of D's (key, value) pairs, as 2-tuples

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.iteritems()` → an iterator over the (key, value) items of D

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.iterkeys()` → an iterator over the keys of D

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.itervalues()` → an iterator over the values of D

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.keys()` → list of D's keys

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.pop(k[, d])` → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.popitem()` → (k, v), remove and return some (key, value) pair as a

2-tuple; but raise `KeyError` if D is empty.

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.update([E], **F)` → None. Update D from dict/iterable

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.values()` → list of D's values

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.viewitems()` → a set-like object providing a view on D's items

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.viewkeys()` → a set-like object providing a view on D's keys

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.viewvalues()` → an object providing a view on D's values

Inherited from `__builtin__.dict`

Special Methods

`NumberedChromaticPitchClassVector.__cmp__(y)` <==> *cmp*(x, y)

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.__contains__(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.__delitem__(*args)`

Inherited from `datastructuretools.ImmutableDictionary`

`NumberedChromaticPitchClassVector.__eq__()`

`x.__eq__(y)` <==> `x==y`

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.__ge__()`

`x.__ge__(y)` <==> `x>=y`

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.__getitem__()`

`x.__getitem__(y)` <==> `x[y]`

Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.__gt__()`

`x.__gt__(y)` <==> `x>y`

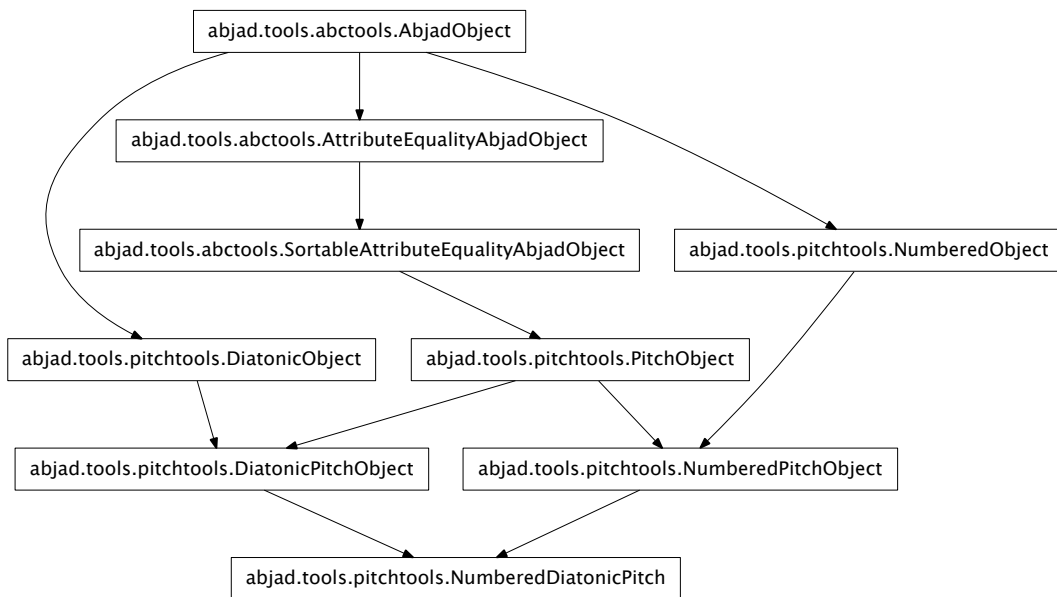
Inherited from `__builtin__.dict`

```

NumberedChromaticPitchClassVector.__iter__() <==> iter(x)
    Inherited from __builtin__.dict
NumberedChromaticPitchClassVector.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.dict
NumberedChromaticPitchClassVector.__len__() <==> len(x)
    Inherited from __builtin__.dict
NumberedChromaticPitchClassVector.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.dict
NumberedChromaticPitchClassVector.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.dict
NumberedChromaticPitchClassVector.__repr__()
NumberedChromaticPitchClassVector.__setitem__(*args)
    Inherited from datastructuretools.ImmutableDictionary
NumberedChromaticPitchClassVector.__str__()

```

pitchtools.NumberedDiatonicPitch



```

class pitchtools.NumberedDiatonicPitch(arg)
    New in version 2.0. Abjad model of a numbered diatonic pitch:

```

```
>>> pitchtools.NumberedDiatonicPitch(7)
NumberedDiatonicPitch(7)
```

Numbered diatonic pitches are immutable.

Read-only Properties

`NumberedDiatonicPitch.chromatic_pitch_number`

Read-only chromatic pitch number:

```
>>> pitchtools.NumberedDiatonicPitch(7).chromatic_pitch_number
12
```

Return integer.

`NumberedDiatonicPitch.diatonic_pitch_number`

Read-only diatonic pitch number:

```
>>> pitchtools.NumberedDiatonicPitch(7).diatonic_pitch_number
7
```

Return integer.

`NumberedDiatonicPitch.named_diatonic_pitch`

Read-only named diatonic pitch:

```
>>> pitchtools.NumberedDiatonicPitch(7).named_diatonic_pitch
NamedDiatonicPitch('c')
```

Return named diatonic pitch.

`NumberedDiatonicPitch.named_diatonic_pitch_class`

Read-only named diatonic pitch-class:

```
>>> pitchtools.NumberedDiatonicPitch(7).named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

Return named diatonic pitch-class.

`NumberedDiatonicPitch.numbered_diatonic_pitch_class`

Read-only numbered diatonic pitch-class:

```
>>> pitchtools.NumberedDiatonicPitch(7).numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

Return numbered diatonic pitch-class.

`NumberedDiatonicPitch.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`NumberedDiatonicPitch.__abs__()`

`NumberedDiatonicPitch.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

NumberedDiatonicPitch.__float__()
Inherited from pitchtools.DiatonicPitchObject

NumberedDiatonicPitch.__ge__(arg)
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from abctools.SortableAttributeEqualityAbjadObject

NumberedDiatonicPitch.__gt__(arg)
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from abctools.SortableAttributeEqualityAbjadObject

NumberedDiatonicPitch.__hash__()
Inherited from pitchtools.PitchObject

NumberedDiatonicPitch.__int__()
Inherited from pitchtools.DiatonicPitchObject

NumberedDiatonicPitch.__le__(arg)
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from abctools.SortableAttributeEqualityAbjadObject

NumberedDiatonicPitch.__lt__(arg)
Initialize new object from *arg* and evaluate comparison attributes.

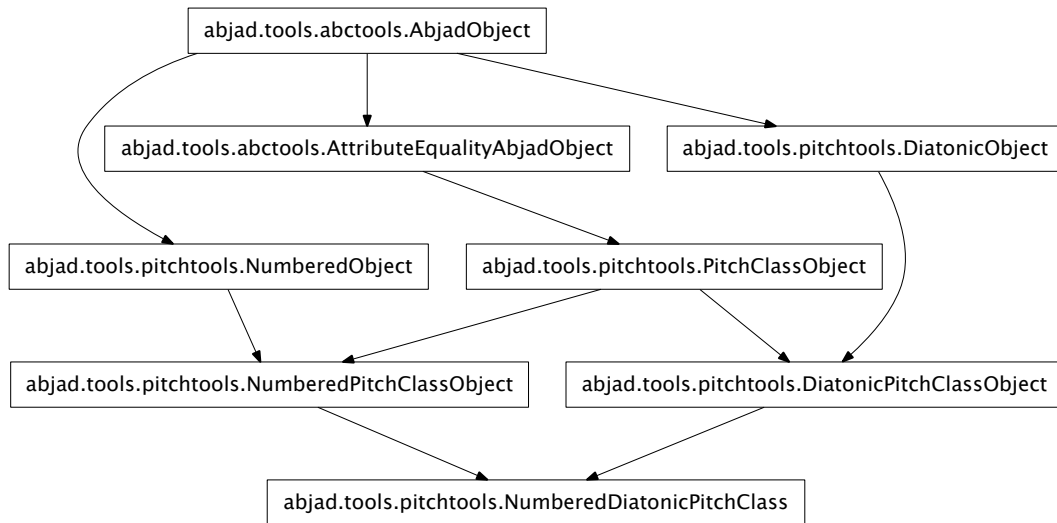
Return boolean.

Inherited from abctools.SortableAttributeEqualityAbjadObject

NumberedDiatonicPitch.__ne__(arg)
Inherited from pitchtools.PitchObject

NumberedDiatonicPitch.__repr__()

NumberedDiatonicPitch.__str__()

pitchtools.NumberedDiatonicPitchClass

class `pitchtools.NumberedDiatonicPitchClass` (*arg*)
 New in version 2.0. Abjad model of a numbered diatonic pitch-class:

```
>>> pitchtools.NumberedDiatonicPitchClass(0)
NumberedDiatonicPitchClass(0)
```

Numbered diatonic pitch-classes are immutable.

Read-only Properties

`NumberedDiatonicPitchClass.named_diatonic_pitch_class`
 Read-only named diatonic pitch-class from numbered diatonic pitch-class:

```
>>> numbered_diatonic_pitch_class = pitchtools.NumberedDiatonicPitchClass(0)
>>> numbered_diatonic_pitch_class.named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

Return named diatonic pitch-class.

`NumberedDiatonicPitchClass.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`NumberedDiatonicPitchClass.__abs__()`
 Inherited from `pitchtools.DiatonicPitchClassObject`

`NumberedDiatonicPitchClass.__eq__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedDiatonicPitchClass.__float__()`

Inherited from `pitchtools.DiatonicPitchClassObject`

`NumberedDiatonicPitchClass.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedDiatonicPitchClass.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`NumberedDiatonicPitchClass.__hash__()`

Inherited from `pitchtools.PitchClassObject`

`NumberedDiatonicPitchClass.__int__()`

Inherited from `pitchtools.DiatonicPitchClassObject`

`NumberedDiatonicPitchClass.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedDiatonicPitchClass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NumberedDiatonicPitchClass.__ne__(arg)`

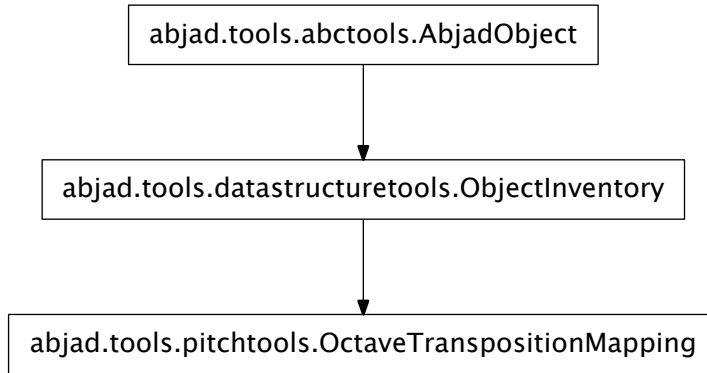
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedDiatonicPitchClass.__repr__()`

`NumberedDiatonicPitchClass.__str__()`

pitchtools.OctaveTranspositionMapping

class `pitchtools.OctaveTranspositionMapping` (*tokens=None, name=None*)

New in version 2.8. Octave transposition mapping:

```
>>> pitchtools.OctaveTranspositionMapping([('A0, C4)', 15), ('C4, C8)', 27])
OctaveTranspositionMapping([('A0, C4)', 15), ('C4, C8)', 27])
```

Octave transposition mappings model `pitchtools.transpose_chromatic_pitch_number_by_octave_transpo` input.

Octave transposition mappings implement the list interface and are mutable.

Read-only Properties

`OctaveTranspositionMapping.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`OctaveTranspositionMapping.name`

Read / write name of inventory.

Inherited from `datastructuretools.ObjectInventory`

Methods

`OctaveTranspositionMapping.append` (*token*)

Change *token* to item and append.

Inherited from `datastructuretools.ObjectInventory`

`OctaveTranspositionMapping.count` (*value*) → integer – return number of occurrences of *value*

Inherited from `__builtin__.list`

OctaveTranspositionMapping.**extend**(*tokens*)
 Change *tokens* to items and extend.
 Inherited from `datastructuretools.ObjectInventory`

OctaveTranspositionMapping.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of value.
 Raises `ValueError` if the value is not present.
 Inherited from `__builtin__.list`

OctaveTranspositionMapping.**insert**()
 L.insert(index, object) – insert object before index
 Inherited from `__builtin__.list`

OctaveTranspositionMapping.**pop**([*index*]) → item – remove and return item at index (default last).
 Raises `IndexError` if list is empty or index is out of range.
 Inherited from `__builtin__.list`

OctaveTranspositionMapping.**remove**()
 L.remove(value) – remove first occurrence of value. Raises `ValueError` if the value is not present.
 Inherited from `__builtin__.list`

OctaveTranspositionMapping.**reverse**()
 L.reverse() – reverse *IN PLACE*
 Inherited from `__builtin__.list`

OctaveTranspositionMapping.**sort**()
 L.sort(cmp=None, key=None, reverse=False) – stable sort *IN PLACE*; cmp(x, y) -> -1, 0, 1
 Inherited from `__builtin__.list`

Special Methods

OctaveTranspositionMapping.**__add__**()
 x.__add__(y) <==> x+y
 Inherited from `__builtin__.list`

OctaveTranspositionMapping.**__contains__**(*token*)
 Inherited from `datastructuretools.ObjectInventory`

OctaveTranspositionMapping.**__delitem__**()
 x.__delitem__(y) <==> del x[y]
 Inherited from `__builtin__.list`

OctaveTranspositionMapping.**__delslice__**()
 x.__delslice__(i, j) <==> del x[i:j]
 Use of negative indices is not supported.
 Inherited from `__builtin__.list`

OctaveTranspositionMapping.**__eq__**()
 x.__eq__(y) <==> x==y
 Inherited from `__builtin__.list`

OctaveTranspositionMapping.**__ge__**()
 x.__ge__(y) <==> x>=y
 Inherited from `__builtin__.list`

```

OctaveTranspositionMapping.__getitem__()
    x.__getitem__(y) <==> x[y]

    Inherited from __builtin__.list

OctaveTranspositionMapping.__getslice__()
    x.__getslice__(i, j) <==> x[i:j]

    Use of negative indices is not supported.

    Inherited from __builtin__.list

OctaveTranspositionMapping.__gt__()
    x.__gt__(y) <==> x>y

    Inherited from __builtin__.list

OctaveTranspositionMapping.__iadd__()
    x.__iadd__(y) <==> x+=y

    Inherited from __builtin__.list

OctaveTranspositionMapping.__imul__()
    x.__imul__(y) <==> x*=y

    Inherited from __builtin__.list

OctaveTranspositionMapping.__iter__() <==> iter(x)
    Inherited from __builtin__.list

OctaveTranspositionMapping.__le__()
    x.__le__(y) <==> x<=y

    Inherited from __builtin__.list

OctaveTranspositionMapping.__len__() <==> len(x)
    Inherited from __builtin__.list

OctaveTranspositionMapping.__lt__()
    x.__lt__(y) <==> x<y

    Inherited from __builtin__.list

OctaveTranspositionMapping.__mul__()
    x.__mul__(n) <==> x*n

    Inherited from __builtin__.list

OctaveTranspositionMapping.__ne__()
    x.__ne__(y) <==> x!=y

    Inherited from __builtin__.list

OctaveTranspositionMapping.__repr__()

OctaveTranspositionMapping.__reversed__()
    L.__reversed__() – return a reverse iterator over the list

    Inherited from __builtin__.list

OctaveTranspositionMapping.__rmul__()
    x.__rmul__(n) <==> n*x

    Inherited from __builtin__.list

```

OctaveTranspositionMapping.__setitem__()

x.__setitem__(i, y) <==> x[i]=y

Inherited from __builtin__.list

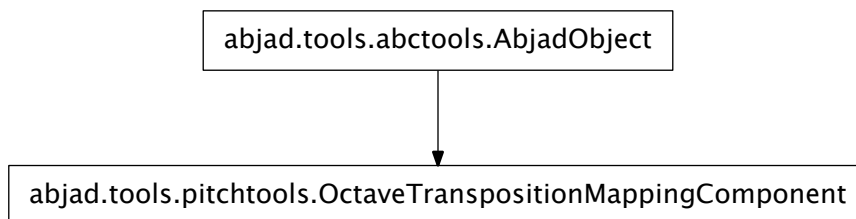
OctaveTranspositionMapping.__setslice__()

x.__setslice__(i, j, y) <==> x[i:j]=y

Use of negative indices is not supported.

Inherited from __builtin__.list

pitchtools.OctaveTranspositionMappingComponent



class pitchtools.OctaveTranspositionMappingComponent(*args)

New in version 2.8. Octave transposition mapping component:

```
>>> pitchtools.OctaveTranspositionMappingComponent(' [A0, C8]', 15)
OctaveTranspositionMappingComponent(' [A0, C8]', 15)
```

Initialize from input parameters separately, from a pair, from a string or from another mapping component.

Model pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping input part. (See the docs for that function.)

Octave transposition mapping components are mutable.

Read-only Properties

OctaveTranspositionMappingComponent.storage_format

Storage format of Abjad object.

Return string.

Inherited from abctools.AbjadObject

Read/write Properties

OctaveTranspositionMappingComponent.source_pitch_range

Read / write source pitch range:

```
>>> mapping_component = pitchtools.OctaveTranspositionMappingComponent (
...     ' [A0, C8]', 15)
>>> mapping_component.source_pitch_range
PitchRange(' [A0, C8]')
```

Return pitch range or none.

`OctaveTranspositionMappingComponent.target_octave_start_pitch`

Read / write target octave start pitch:

```
>>> mapping_component = pitchtools.OctaveTranspositionMappingComponent (
...     '[A0, C8]', 15)
>>> mapping_component.target_octave_start_pitch
NumberedChromaticPitch(15)
```

Return numbered chromatic pitch or none.

Special Methods

`OctaveTranspositionMappingComponent.__eq__(other)`

`OctaveTranspositionMappingComponent.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OctaveTranspositionMappingComponent.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OctaveTranspositionMappingComponent.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OctaveTranspositionMappingComponent.__lt__(arg)`

Abjad objects by default do not implement this method.

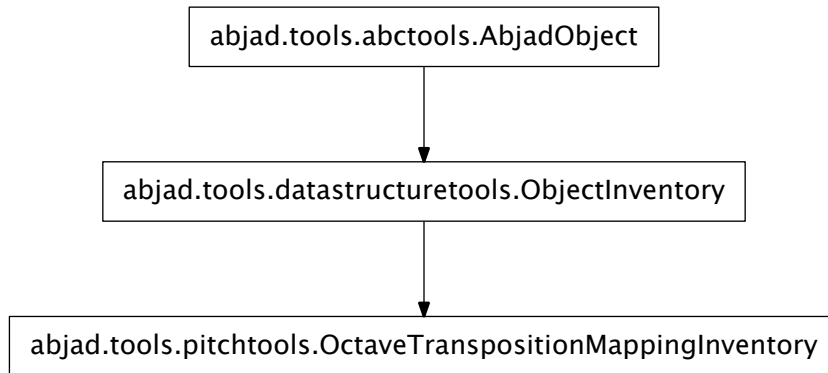
Raise exception.

Inherited from `abctools.AbjadObject`

`OctaveTranspositionMappingComponent.__ne__(other)`

`OctaveTranspositionMappingComponent.__repr__()`

pitchtools.OctaveTranspositionMappingInventory



class `pitchtools.OctaveTranspositionMappingInventory` (*tokens=None, name=None*)

New in version 2.8. Model of an ordered list of octave transposition mappings:

```

>>> mapping_1 = pitchtools.OctaveTranspositionMapping([('A0, C4)', 15], ('C4, C8)', 27)])
>>> mapping_2 = pitchtools.OctaveTranspositionMapping([('A0, C8]', -18)])
>>> inventory = pitchtools.OctaveTranspositionMappingInventory([mapping_1, mapping_2])

>>> z(inventory)
pitchtools.OctaveTranspositionMappingInventory([
  pitchtools.OctaveTranspositionMapping([
    pitchtools.OctaveTranspositionMappingComponent(
      pitchtools.PitchRange(
        '[A0, C4]'
      ),
      pitchtools.NumberedChromaticPitch(
        15
      )
    ),
    pitchtools.OctaveTranspositionMappingComponent(
      pitchtools.PitchRange(
        '[C4, C8]'
      ),
      pitchtools.NumberedChromaticPitch(
        27
      )
    )
  ]),
  pitchtools.OctaveTranspositionMapping([
    pitchtools.OctaveTranspositionMappingComponent(
      pitchtools.PitchRange(
        '[A0, C8]'
      ),
      pitchtools.NumberedChromaticPitch(
        -18
      )
    )
  ])
])
  
```



```
)
    ])
])
```

Octave transposition mapping inventories implement list interface and are mutable.

Read-only Properties

`OctaveTranspositionMappingInventory.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`OctaveTranspositionMappingInventory.name`

Read / write name of inventory.

Inherited from `datastructuretools.ObjectInventory`

Methods

`OctaveTranspositionMappingInventory.append(token)`

Change *token* to item and append.

Inherited from `datastructuretools.ObjectInventory`

`OctaveTranspositionMappingInventory.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`OctaveTranspositionMappingInventory.extend(tokens)`

Change *tokens* to items and extend.

Inherited from `datastructuretools.ObjectInventory`

`OctaveTranspositionMappingInventory.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`OctaveTranspositionMappingInventory.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`OctaveTranspositionMappingInventory.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`OctaveTranspositionMappingInventory.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`OctaveTranspositionMappingInventory.reverse()`

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**sort**()
L.sort(cmp=None, key=None, reverse=False) – stable sort *IN PLACE*; cmp(x, y) -> -1, 0, 1
Inherited from `__builtin__.list`

Special Methods

OctaveTranspositionMappingInventory.**__add__**()
x.**__add__**(y) <==> x+y
Inherited from `__builtin__.list`
OctaveTranspositionMappingInventory.**__contains__**(token)
Inherited from `datastructuretools.ObjectInventory`

OctaveTranspositionMappingInventory.**__delitem__**()
x.**__delitem__**(y) <==> del x[y]
Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__delslice__**()
x.**__delslice__**(i, j) <==> del x[i:j]
Use of negative indices is not supported.
Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__eq__**()
x.**__eq__**(y) <==> x==y
Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__ge__**()
x.**__ge__**(y) <==> x>=y
Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__getitem__**()
x.**__getitem__**(y) <==> x[y]
Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__getslice__**()
x.**__getslice__**(i, j) <==> x[i:j]
Use of negative indices is not supported.
Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__gt__**()
x.**__gt__**(y) <==> x>y
Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__iadd__**()
x.**__iadd__**(y) <==> x+=y
Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__imul__**()
x.**__imul__**(y) <==> x*=y
Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__iter__**() <==> iter(x)
Inherited from `__builtin__.list`

```

OctaveTranspositionMappingInventory.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.list

OctaveTranspositionMappingInventory.__len__() <==> len(x)
    Inherited from __builtin__.list

OctaveTranspositionMappingInventory.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.list

OctaveTranspositionMappingInventory.__mul__()
    x.__mul__(n) <==> x*n
    Inherited from __builtin__.list

OctaveTranspositionMappingInventory.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.list

OctaveTranspositionMappingInventory.__repr__()
    Inherited from datastructuretools.ObjectInventory

OctaveTranspositionMappingInventory.__reversed__()
    L.__reversed__() – return a reverse iterator over the list
    Inherited from __builtin__.list

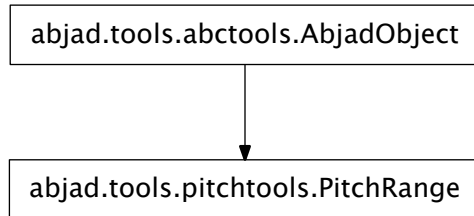
OctaveTranspositionMappingInventory.__rmul__()
    x.__rmul__(n) <==> n*x
    Inherited from __builtin__.list

OctaveTranspositionMappingInventory.__setitem__()
    x.__setitem__(i, y) <==> x[i]=y
    Inherited from __builtin__.list

OctaveTranspositionMappingInventory.__setslice__()
    x.__setslice__(i, j, y) <==> x[i:j]=y
    Use of negative indices is not supported.
    Inherited from __builtin__.list

```

pitchtools.PitchRange



class `pitchtools.PitchRange` (*args, **kwargs)

New in version 2.0. Abjad model of pitch range:

```
>>> pitchtools.PitchRange(-12, 36)
PitchRange(' [C3, C7]')
```

Initialize from pitch numbers, pitch names, pitch instances, one-line reprs or other pitch range objects.

Pitch ranges test for equality and inequality against other pitch ranges.

Pitch ranges test less than, greater than, less-equal and greater-equal against pitches.

Pitch ranges do not sort relative to other pitch ranges.

Pitch ranges are immutable.

Read-only Properties

`PitchRange.one_line_named_chromatic_pitch_repr`

Read-only one-line named chromatic pitch repr of pitch of range:

```
>>> pitch_range = pitchtools.PitchRange(-12, 36)
>>> pitch_range.one_line_named_chromatic_pitch_repr
'[C3, C7]'
```

Return string.

`PitchRange.one_line_numbered_chromatic_pitch_repr`

Read-only one-line numbered chromatic pitch repr of pitch of range:

```
>>> pitch_range = pitchtools.PitchRange(-12, 36)
>>> pitch_range.one_line_numbered_chromatic_pitch_repr
'[-12, 36]'
```

Return string.

`PitchRange.pitch_range_name`

New in version 2.7. Read-only name of pitch range:

```
>>> pitch_range = pitchtools.PitchRange(-12, 36, pitch_range_name='four-octave range')
>>> pitch_range.pitch_range_name
'four-octave range'
```

Return string or none.

PitchRange.pitch_range_name_markup

New in version 2.7. Read-only markup of pitch range name:

```
>>> from abjad.tools.markuptools import Markup

>>> pitch_range = pitchtools.PitchRange(-12, 36,
...     pitch_range_name_markup=Markup('four-octave range'))
>>> pitch_range.pitch_range_name_markup
Markup(('four-octave range',))
```

Default to *pitch_range_name* when *pitch_range_name_markup* not set explicitly.

Return markup or none.

PitchRange.start_pitch

Read-only start pitch of range:

```
>>> pitch_range = pitchtools.PitchRange(-12, 36)
>>> pitch_range.start_pitch
NamedChromaticPitch('c')
```

Return pitch.

PitchRange.start_pitch_is_included_in_range

Read-only boolean true when start pitch is included in range. Otherwise false:

```
>>> pitch_range = pitchtools.PitchRange(-12, 36)
>>> pitch_range.start_pitch_is_included_in_range
True
```

Return boolean.

PitchRange.stop_pitch

Read-only stop pitch of range:

```
>>> pitch_range = pitchtools.PitchRange(-12, 36)
>>> pitch_range.stop_pitch
NamedChromaticPitch("c' ' ' ' ")
```

Return pitch.

PitchRange.stop_pitch_is_included_in_range

Read-only boolean true when stop pitch is included in range. Otherwise false:

```
>>> pitch_range = pitchtools.PitchRange(-12, 36)
>>> pitch_range.stop_pitch_is_included_in_range
True
```

Return boolean.

PitchRange.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`PitchRange.__contains__(arg)`

`PitchRange.__eq__(arg)`

`PitchRange.__ge__(arg)`

PitchRange.__gt__(arg)

PitchRange.__le__(arg)

PitchRange.__lt__(arg)

PitchRange.__ne__(arg)

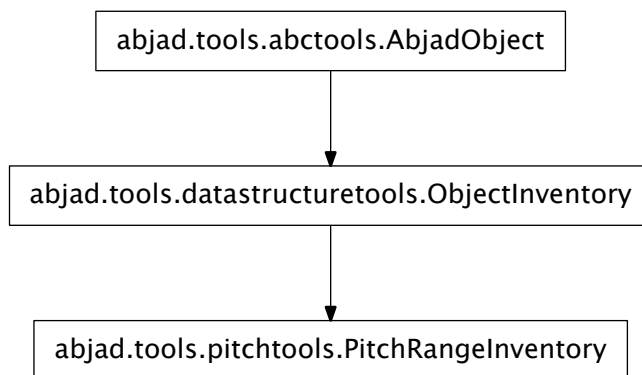
PitchRange.__repr__()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`pitchtools.PitchRangeInventory`



class `pitchtools.PitchRangeInventory` (*tokens=None, name=None*)

New in version 2.7. Abjad model of an ordered list of pitch ranges:

```
>>> pitchtools.PitchRangeInventory(['[C3, C6]', '[C4, C6]'])
PitchRangeInventory([PitchRange('[C3, C6]'), PitchRange('[C4, C6]')])
```

Pitch range inventories implement list interface and are mutable.

Read-only Properties

`PitchRangeInventory.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`PitchRangeInventory.name`

Read / write name of inventory.

Inherited from `datastructuretools.ObjectInventory`

Methods

`PitchRangeInventory.append(token)`

Change *token* to item and append.

Inherited from `datastructuretools.ObjectInventory`

`PitchRangeInventory.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`PitchRangeInventory.extend(tokens)`

Change *tokens* to items and extend.

Inherited from `datastructuretools.ObjectInventory`

`PitchRangeInventory.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`PitchRangeInventory.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`PitchRangeInventory.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`PitchRangeInventory.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`PitchRangeInventory.reverse()`

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`PitchRangeInventory.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`PitchRangeInventory.__add__()`

`x.__add__(y) <==> x+y`

Inherited from `__builtin__.list`

`PitchRangeInventory.__contains__(token)`

Inherited from `datastructuretools.ObjectInventory`

`PitchRangeInventory.__delitem__()`

`x.__delitem__(y) <==> del x[y]`

Inherited from `__builtin__.list`

`PitchRangeInventory.__delslice__()`

`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

```

PitchRangeInventory.__eq__()
    x.__eq__(y) <==> x==y
    Inherited from __builtin__.list

PitchRangeInventory.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.list

PitchRangeInventory.__getitem__()
    x.__getitem__(y) <==> x[y]
    Inherited from __builtin__.list

PitchRangeInventory.__getslice__()
    x.__getslice__(i, j) <==> x[i:j]
    Use of negative indices is not supported.
    Inherited from __builtin__.list

PitchRangeInventory.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.list

PitchRangeInventory.__iadd__()
    x.__iadd__(y) <==> x+=y
    Inherited from __builtin__.list

PitchRangeInventory.__imul__()
    x.__imul__(y) <==> x*=y
    Inherited from __builtin__.list

PitchRangeInventory.__iter__() <==> iter(x)
    Inherited from __builtin__.list

PitchRangeInventory.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.list

PitchRangeInventory.__len__() <==> len(x)
    Inherited from __builtin__.list

PitchRangeInventory.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.list

PitchRangeInventory.__mul__()
    x.__mul__(n) <==> x*n
    Inherited from __builtin__.list

PitchRangeInventory.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.list

PitchRangeInventory.__repr__()
    Inherited from datastructuretools.ObjectInventory

```



```
PitchRangeInventory.__reversed__()
    L.__reversed__() – return a reverse iterator over the list

    Inherited from __builtin__.list

PitchRangeInventory.__rmul__()
    x.__rmul__(n) <==> n*x

    Inherited from __builtin__.list

PitchRangeInventory.__setitem__()
    x.__setitem__(i, y) <==> x[i]=y

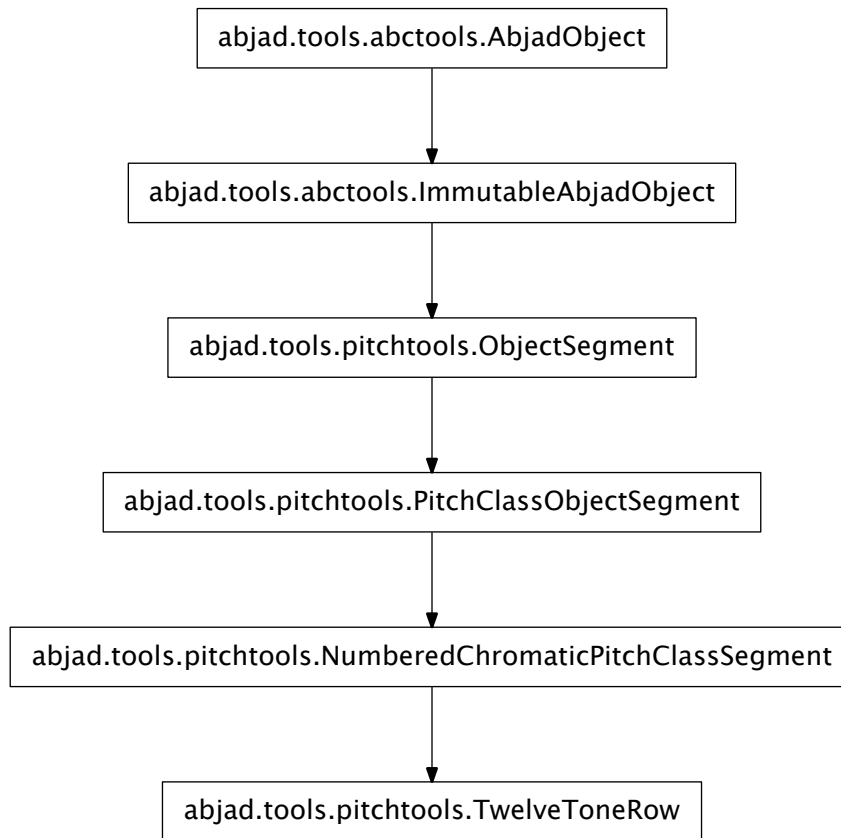
    Inherited from __builtin__.list

PitchRangeInventory.__setslice__()
    x.__setslice__(i, j, y) <==> x[i:j]=y

    Use of negative indices is not supported.

    Inherited from __builtin__.list
```

pitchtools.TwelveToneRow



class `pitchtools.TwelveToneRow(*args, **kwargs)`

New in version 2.0. Abjad model of twelve-tone row:

```
>>> pitchtools.TwelveToneRow([0, 1, 11, 9, 3, 6, 7, 5, 4, 10, 2, 8])
TwelveToneRow([0, 1, 11, 9, 3, 6, 7, 5, 4, 10, 2, 8])
```

Twelve-tone rows validate pitch-classes at initialization.

Twelve-tone rows inherit canonical operators from numbered chromatic pitch-class segment.

Twelve-tone rows return numbered chromatic pitch-class segments on calls to `getslice`.

Twelve-tone rows are immutable.

Read-only Properties

`TwelveToneRow.inversion_equivalent_chromatic_interval_class_segment`

Read-only inversion-equivalent chromatic interval-class segment:

```
>>> segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5, 6, 7, 10.5, 7])

>>> segment.inversion_equivalent_chromatic_interval_class_segment
InversionEquivalentChromaticIntervalClassSegment(0.5, 4.5, 1, 3.5, 3.5)
```

Return inversion-equivalent chromatic interval-class segment.

Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.numbered_chromatic_pitch_class_set`

Read-only numbered chromatic pitch-class set from numbered chromatic pitch-class segment:

```
>>> segment.numbered_chromatic_pitch_class_set
NumberedChromaticPitchClassSet([6, 7, 10, 10.5])
```

Return numbered chromatic pitch-class set.

Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TwelveToneRow.alpha()`

Morris alpha transform of numbered chromatic pitch-class segment:

```
>>> segment.alpha()
NumberedChromaticPitchClassSegment([11, 11.5, 7, 6, 11.5, 6])
```

Return numbered chromatic pitch-class segment.

Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.count(value) → integer` – return number of occurrences of value

Inherited from `__builtin__.tuple`

`TwelveToneRow.index(value[, start[, stop]]) → integer` – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`TwelveToneRow.invert()`

Invert numbered chromatic pitch-class segment:

```
>>> segment.invert()
NumberedChromaticPitchClassSegment([2, 1.5, 6, 5, 1.5, 5])
```

Return numbered chromatic pitch-class segment.

Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.multiply(n)`

Multiply numbered chromatic pitch-class segment by *n*:

```
>>> segment.multiply(5)
NumberedChromaticPitchClassSegment([2, 4.5, 6, 11, 4.5, 11])
```

Return numbered chromatic pitch-class segment.

Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.retrograde()`

Retrograde of numbered chromatic pitch-class segment:

```
>>> segment.retrograde()
NumberedChromaticPitchClassSegment([7, 10.5, 7, 6, 10.5, 10])
```

Return numbered chromatic pitch-class segment.

Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.rotate(n)`

Rotate numbered chromatic pitch-class segment:

```
>>> segment.rotate(1)
NumberedChromaticPitchClassSegment([7, 10, 10.5, 6, 7, 10.5])
```

Return numbered chromatic pitch-class segment.

Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.transpose(n)`

Transpose numbered chromatic pitch-class segment:

```
>>> segment.transpose(10)
NumberedChromaticPitchClassSegment([8, 8.5, 4, 5, 8.5, 5])
```

Return numbered chromatic pitch-class segment.

Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

Special Methods

`TwelveToneRow.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`TwelveToneRow.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`TwelveToneRow.__eq__(arg)`

`TwelveToneRow.__ge__()`

`x.__ge__(y) <==> x>=y`

```

    Inherited from __builtin__.tuple
TwelveToneRow.__getitem__()
    x.__getitem__(y) <==> x[y]

    Inherited from __builtin__.tuple
TwelveToneRow.__getslice__(start, stop)
TwelveToneRow.__gt__()
    x.__gt__(y) <==> x>y

    Inherited from __builtin__.tuple
TwelveToneRow.__hash__() <==> hash(x)
    Inherited from __builtin__.tuple
TwelveToneRow.__iter__() <==> iter(x)
    Inherited from __builtin__.tuple
TwelveToneRow.__le__()
    x.__le__(y) <==> x<=y

    Inherited from __builtin__.tuple
TwelveToneRow.__len__() <==> len(x)
    Inherited from __builtin__.tuple
TwelveToneRow.__lt__()
    x.__lt__(y) <==> x<y

    Inherited from __builtin__.tuple
TwelveToneRow.__mul__(n)
TwelveToneRow.__ne__(arg)
TwelveToneRow.__repr__()
    Inherited from pitchtools.NumberedChromaticPitchClassSegment
TwelveToneRow.__rmul__(n)
TwelveToneRow.__str__()
    Inherited from pitchtools.NumberedChromaticPitchClassSegment

```

functions

`pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs`

`pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs(expr)`

New in version 1.1. True when all elements of *expr* are pitch tokens. Otherwise false:

```

>>> pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs(
... [('c', 4), ('d', 4), pitchtools.NamedChromaticPitch('e', 4)])
True

```

Return boolean.

pitchtools.all_are_named_chromatic_pitch_tokens**pitchtools.all_are_named_chromatic_pitch_tokens** (*expr*)New in version 2.6. True when *expr* is a sequence of named chromatic pitch tokens:

```
>>> named_chromatic_pitch_tokens = [('c', 4), pitchtools.NamedChromaticPitch("a")]
```

```
>>> pitchtools.all_are_named_chromatic_pitch_tokens(named_chromatic_pitch_tokens)
True
```

True when *expr* is an empty sequence:

```
>>> pitchtools.all_are_named_chromatic_pitch_tokens([])
True
```

Otherwise false:

```
>>> pitchtools.all_are_named_chromatic_pitch_tokens('foo')
False
```

Return boolean.

pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string**pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string** (*alphabetic_accidental_abbreviation*)New in version 2.5. Change *alphabetic_accidental_abbreviation* to symbolic accidental string:

```
>>> pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string('tqs')
'#+'
```

None when *alphabetic_accidental_abbreviation* is not a valid alphabetic accidental abbreviation.

Return string or none.

pitchtools.apply_accidental_to_named_chromatic_pitch**pitchtools.apply_accidental_to_named_chromatic_pitch** (*named_chromatic_pitch*, *accidental=None*)New in version 2.0. Apply *accidental* to *named_chromatic_pitch*:

```
>>> pitch = pitchtools.NamedChromaticPitch("cs'")
>>> pitchtools.apply_accidental_to_named_chromatic_pitch(pitch, 'f')
NamedChromaticPitch("c'")
```

Return new named pitch.

pitchtools.calculate_harmonic_chromatic_interval**pitchtools.calculate_harmonic_chromatic_interval** (*pitch_carrier_1*, *pitch_carrier_2*)New in version 2.0. Calculate harmonic chromatic interval from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_harmonic_chromatic_interval(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
HarmonicChromaticInterval(14)
```

Return harmonic chromatic interval.

`pitchtools.calculate_harmonic_chromatic_interval_class`

`pitchtools.calculate_harmonic_chromatic_interval_class` (*pitch_carrier_1*,
pitch_carrier_2)

New in version 2.0. Calculate harmonic chromatic interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_harmonic_chromatic_interval_class(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
HarmonicChromaticIntervalClass(2)
```

Return harmonic chromatic interval-class.

`pitchtools.calculate_harmonic_counterpoint_interval`

`pitchtools.calculate_harmonic_counterpoint_interval` (*pitch_carrier_1*,
pitch_carrier_2)

New in version 2.0. Calculate harmonic counterpoint interval *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_harmonic_counterpoint_interval(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
HarmonicCounterpointInterval(9)
```

Return harmonic counterpoint interval-class.

`pitchtools.calculate_harmonic_counterpoint_interval_class`

`pitchtools.calculate_harmonic_counterpoint_interval_class` (*pitch_carrier_1*,
pitch_carrier_2)

New in version 2.0. Calculate harmonic counterpoint interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_harmonic_counterpoint_interval_class(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
HarmonicCounterpointIntervalClass(2)
```

Return harmonic counterpoint interval-class.

`pitchtools.calculate_harmonic_diatonic_interval`

`pitchtools.calculate_harmonic_diatonic_interval` (*pitch_carrier_1*, *pitch_carrier_2*)

New in version 2.0. Calculate harmonic diatonic interval from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_harmonic_diatonic_interval(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
HarmonicDiatonicInterval('M9')
```

Return harmonic diatonic interval.

`pitchtools.calculate_harmonic_diatonic_interval_class`

`pitchtools.calculate_harmonic_diatonic_interval_class` (*pitch_carrier_1*,
pitch_carrier_2)

New in version 2.0. Calculate harmonic diatonic interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_harmonic_diatonic_interval_class(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
HarmonicDiatonicIntervalClass('M2')
```

Return harmonic diatonic interval-class.

`pitchtools.calculate_melodic_chromatic_interval`

`pitchtools.calculate_melodic_chromatic_interval` (*pitch_carrier_1*, *pitch_carrier_2*)
 New in version 2.0. Calculate melodic chromatic interval from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_melodic_chromatic_interval(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
MelodicChromaticInterval(+14)
```

Return melodic chromatic interval.

`pitchtools.calculate_melodic_chromatic_interval_class`

`pitchtools.calculate_melodic_chromatic_interval_class` (*pitch_carrier_1*,
 pitch_carrier_2)
 New in version 2.0. Calculate melodic chromatic interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_melodic_chromatic_interval_class(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
MelodicChromaticIntervalClass(+2)
```

Return melodic chromatic interval-class.

`pitchtools.calculate_melodic_counterpoint_interval`

`pitchtools.calculate_melodic_counterpoint_interval` (*pitch_carrier_1*, *pitch_carrier_2*)
 New in version 2.0. Calculate melodic counterpoint interval *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_melodic_counterpoint_interval(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
MelodicCounterpointInterval(+9)
```

Return melodic counterpoint interval.

`pitchtools.calculate_melodic_counterpoint_interval_class`

`pitchtools.calculate_melodic_counterpoint_interval_class` (*pitch_carrier_1*,
 pitch_carrier_2)
 New in version 2.0. Calculate melodic counterpoint interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_melodic_counterpoint_interval_class(
... pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
MelodicCounterpointIntervalClass(+2)
```

Return melodic counterpoint interval-class.

`pitchtools.calculate_melodic_diatonic_interval`

`pitchtools.calculate_melodic_diatonic_interval` (*pitch_carrier_1*, *pitch_carrier_2*)

New in version 2.0. Calculate melodic diatonic interval from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_melodic_diatonic_interval(
...     pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
MelodicDiatonicInterval('+M9')
```

Return melodic diatonic interval.

`pitchtools.calculate_melodic_diatonic_interval_class`

`pitchtools.calculate_melodic_diatonic_interval_class` (*pitch_carrier_1*,
pitch_carrier_2)

New in version 2.0. Calculate melodic diatonic interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
>>> pitchtools.calculate_melodic_diatonic_interval_class(
...     pitchtools.NamedChromaticPitch(-2), pitchtools.NamedChromaticPitch(12))
MelodicDiatonicIntervalClass('+M2')
```

Return melodic diatonic interval-class.

`pitchtools.chromatic_pitch_class_name_to_chromatic_pitch_class_number`

`pitchtools.chromatic_pitch_class_name_to_chromatic_pitch_class_number` (*chromatic_pitch_class_name*)

New in version 2.0. Change *chromatic_pitch_class_name* to chromatic pitch-class number:

```
>>> pitchtools.chromatic_pitch_class_name_to_chromatic_pitch_class_number('cs')
1
```

Return chromatic pitch-class number.

`pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name`

`pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name` (*chromatic_pitch_class_name*)

New in version 2.0. Change *chromatic_pitch_class_name* to diatonic pitch-class name:

```
>>> pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name('cs')
'c'
```

Return string.

`pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name`

`pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name` (*chromatic_pitch_class_number*)

New in version 1.1. Change *chromatic_pitch_class_number* to chromatic pitch-class name:

```
>>> tmp = pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name
>>> for n in range(0, 13):
...     pc = n / 2.0
...     pitch_name = tmp(pc)
...     print '%s %s' % (pc, pitch_name)
... 
```



```

0.0  c
0.5  cqs
1.0  cs
1.5  dqf
2.0  d
2.5  dqs
3.0  ef
3.5  eqf
4.0  e
4.5  eqs
5.0  f
5.5  fqs
6.0  fs

```

Return string. Changed in version 2.0: renamed `pitchtools.pc_to_pitch_name()` to `pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name()`.

`pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_flats`

`pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_flats` (*chromatic_pitch_class_number*)

New in version 1.1. Change chromatic pitch-class number to chromatic pitch-class name with flats:

```

>>> tmp = pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_flats

>>> for n in range(13):
...     pc = n / 2.0
...     name = tmp(pc)
...     print '%s %s' % (pc, name)
...
0.0  c
0.5  dtqf
1.0  df
1.5  dqf
2.0  d
2.5  etqf
3.0  ef
3.5  eqf
4.0  e
4.5  fqf
5.0  f
5.5  gtqf
6.0  gf

```

Return string. Changed in version 2.0: renamed `pitchtools.pc_to_pitch_name_flats()` to `pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_flats()`.

`pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_sharps`

`pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_sharps` (*chromatic_pitch_class_number*)

New in version 1.1. Change *chromatic_pitch_class_number* to chromatic pitch-class name with sharps:

```

>>> tmp = pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_sharps
>>> for n in range(13):
...     pc = n / 2.0
...     name = tmp(pc)
...     print '%s %s' % (pc, name)

```

```

...
0.0   c
0.5   cqs
1.0   cs
1.5   ctqs
2.0   d
2.5   dqs
3.0   ds
3.5   dtqs
4.0   e
4.5   eqs
5.0   f
5.5   fqs
6.0   fs

```

Return string. Changed in version 2.0: renamed `pitchtools.pc_to_pitch_name_sharps()` to `pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_sharps()`.

`pitchtools.chromatic_pitch_class_number_to_diatonic_pitch_class_number`

`pitchtools.chromatic_pitch_class_number_to_diatonic_pitch_class_number` (*chromatic_pitch_class_number*)
 New in version 2.0. Change *chromatic_pitch_class_number* to diatonic pitch-class number:

```
>>> pitchtools.chromatic_pitch_class_number_to_diatonic_pitch_class_number(1)
0
```

Return integer.

`pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_name`

`pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_name` (*chromatic_pitch_name*)
 New in version 2.0. Change *chromatic_pitch_name* to chromatic pitch-class name:

```
>>> pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_name("cs'")
'cs'
```

Return string.

`pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_number`

`pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_number` (*chromatic_pitch_name*)
 New in version 2.0. Change *chromatic_class_name* to chromatic pitch-class-number:

```
>>> pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_number("cs'")
1
```

Return integer or float.

`pitchtools.chromatic_pitch_name_to_chromatic_pitch_number`

`pitchtools.chromatic_pitch_name_to_chromatic_pitch_number` (*chromatic_pitch_name*)
 New in version 2.0. Change *chromatic_pitch_name* to chromatic pitch number:

```
>>> pitchtools.chromatic_pitch_name_to_chromatic_pitch_number("cs' ' ")
13
```

Return integer or float.

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_name`

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_name(chromatic_pitch_name)`
New in version 2.0. Change *chromatic_pitch_name* to diatonic pitch name:

```
>>> pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_name("cs' ' ")
'c'
```

Return string.

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_number`

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_number(chromatic_pitch_name)`
New in version 2.0. Change *chromatic_pitch_name* to diatonic pitch-class number:

```
>>> pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_number("cs' ' ")
0
```

Return integer.

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_name`

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_name(chromatic_pitch_name)`
New in version 2.0. Change *chromatic_pitch_name* to diatonic pitch name:

```
>>> pitchtools.chromatic_pitch_name_to_diatonic_pitch_name("cs' ' ")
"c' ' "
```

Return string.

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_number`

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_number(chromatic_pitch_name)`
New in version 2.0. Change *chromatic_pitch_name* to diatonic pitch number:

```
>>> pitchtools.chromatic_pitch_name_to_diatonic_pitch_number("cs' ' ")
7
```

Return integer.

`pitchtools.chromatic_pitch_name_to_octave_number`

`pitchtools.chromatic_pitch_name_to_octave_number(chromatic_pitch_name)`
New in version 2.0. Change *chromatic_pitch_name* to octave number:

```
>>> pitchtools.chromatic_pitch_name_to_octave_number('cs')
3
```

Return integer.

`pitchtools.chromatic_pitch_names_string_to_named_chromatic_pitch_list`

`pitchtools.chromatic_pitch_names_string_to_named_chromatic_pitch_list` (*chromatic_pitch_names_string*)
 New in version 2.0. Change *chromatic_pitch_names_string* to named chromatic pitch list:

```
>>> string = "cs, cs cs' cs'"
>>> result = pitchtools.chromatic_pitch_names_string_to_named_chromatic_pitch_list(string)

>>> for named_chromatic_pitch in result:
...     named_chromatic_pitch
...
NamedChromaticPitch('cs,')
NamedChromaticPitch('cs')
NamedChromaticPitch("cs' ")
NamedChromaticPitch("cs' ' ")
```

Return list of named chromatic pitches.

`pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number`

`pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number` (*chromatic_pitch_number*,
accidental_semitones)

New in version 1.1. Change *chromatic_pitch_number* and *accidental_semitones* to octave number:

```
>>> pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number(12, -2)
5
```

Return integer. Changed in version 2.0: renamed `pitchtools.pitch_number_and_accidental_semitones_to_octave_number` to `pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number()`.

`pitchtools.chromatic_pitch_number_to_chromatic_pitch_class_number`

`pitchtools.chromatic_pitch_number_to_chromatic_pitch_class_number` (*chromatic_pitch_number*)
 New in version 2.0. Change *chromatic_pitch_number* to chromatic pitch-class number:

```
>>> pitchtools.chromatic_pitch_number_to_chromatic_pitch_class_number(13)
1
```

Return integer or float.

`pitchtools.chromatic_pitch_number_to_chromatic_pitch_name`

`pitchtools.chromatic_pitch_number_to_chromatic_pitch_name` (*chromatic_pitch_number*,
accidental_spelling='mixed')

New in version 2.0. Change *chromatic_pitch_number* to chromatic pitch name:

```
>>> pitchtools.chromatic_pitch_number_to_chromatic_pitch_name(13)
"cs' ' "
```

Return string.

pitchtools.chromatic_pitch_number_to_chromatic_pitch_triple

`pitchtools.chromatic_pitch_number_to_chromatic_pitch_triple` (*chromatic_pitch_number*,
accidental_spelling=*'mixed'*)

Change *chromatic_pitch_number* to diatonic pitch-class name / alphabetic accidental abbreviation / octave number triple:

```
>>> pitchtools.chromatic_pitch_number_to_chromatic_pitch_triple(
... 13, accidental_spelling='sharps')
('c', Accidental('s'), 5)
```

Return tuple. Changed in version 2.0: renamed `pitchtools.number_to_letter_accidental_octave()` to `pitchtools.chromatic_pitch_number_to_chromatic_pitch_triple()`.

pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_number

`pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_number` (*chromatic_pitch_number*)
 New in version 2.0. Change *chromatic_pitch_number* to diatonic pitch-class number:

```
>>> pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_number(13)
0
```

Return integer.

pitchtools.chromatic_pitch_number_to_diatonic_pitch_number

`pitchtools.chromatic_pitch_number_to_diatonic_pitch_number` (*chromatic_pitch_number*)
 New in version 2.0. Change *chromatic_pitch_number* to diatonic pitch number:

```
>>> pitchtools.chromatic_pitch_number_to_diatonic_pitch_number(13)
7
```

Return integer.

pitchtools.chromatic_pitch_number_to_octave_number

`pitchtools.chromatic_pitch_number_to_octave_number` (*chromatic_pitch_number*)
 New in version 1.1. Change *chromatic_pitch_number* to octave number:

```
>>> pitchtools.chromatic_pitch_number_to_octave_number(13)
5
```

Return integer. Changed in version 2.0: renamed `pitchtools.pitch_number_to_octave()` to `pitchtools.chromatic_pitch_number_to_octave_number()`.

pitchtools.clef_and_staff_position_number_to_named_chromatic_pitch

`pitchtools.clef_and_staff_position_number_to_named_chromatic_pitch` (*clef*,
staff_position_number)

New in version 2.0. Change *clef* and *staff_position_number* to named chromatic pitch:

```
>>> clef = contexttools.ClefMark('treble')
>>> for n in range(-6, 6):
...     pitch = pitchtools.clef_and_staff_position_number_to_named_chromatic_pitch(clef, n)
...     print '%s\t%s\t%s' % (clef.clef_name, n, pitch)
treble    -6 c'
treble    -5 d'
treble    -4 e'
treble    -3 f'
treble    -2 g'
treble    -1 a'
treble     0 b'
treble     1 c''
treble     2 d''
treble     3 e''
treble     4 f''
treble     5 g''
```

Return named chromatic pitch.

pitchtools.contains_subsegment

pitchtools.contains_subsegment (*chromatic_pitch_class_numbers*, *chromatic_pitch_numbers*)
 New in version 1.1. True when *chromatic_pitch_numbers* contain *chromatic_pitch_class_numbers* as subsegment:

```
>>> pcs = [2, 7, 10]
>>> pitches = [6, 9, 12, 13, 14, 19, 22, 27, 28, 29, 32, 35]
>>> pitchtools.contains_subsegment(pcs, pitches)
True
```

Return boolean.

pitchtools.diatonic_pitch_class_name_to_chromatic_pitch_class_number

pitchtools.diatonic_pitch_class_name_to_chromatic_pitch_class_number (*diatonic_pitch_class_name*)
 New in version 1.1. Change *diatonic_pitch_class_name* to chromatic pitch-class number:

```
>>> pitchtools.diatonic_pitch_class_name_to_chromatic_pitch_class_number('f')
5
```

Return integer.

pitchtools.diatonic_pitch_class_name_to_diatonic_pitch_class_number

pitchtools.diatonic_pitch_class_name_to_diatonic_pitch_class_number (*diatonic_pitch_class_name*)
 New in version 2.0. Change *diatonic_pitch_class_name* to diatonic pitch-class number:

```
>>> pitchtools.diatonic_pitch_class_name_to_diatonic_pitch_class_number('c')
0
```

Return integer.

pitchtools.diatonic_pitch_class_number_to_chromatic_pitch_class_number

`pitchtools.diatonic_pitch_class_number_to_chromatic_pitch_class_number` (*diatonic_pitch_class_number*)

New in version 2.0. Change *diatonic_pitch_class_number* to chromatic pitch-class number:

```
>>> pitchtools.diatonic_pitch_class_number_to_chromatic_pitch_class_number(6)
11
```

Return nonnegative integer.

pitchtools.diatonic_pitch_class_number_to_diatonic_pitch_class_name

`pitchtools.diatonic_pitch_class_number_to_diatonic_pitch_class_name` (*diatonic_pitch_class_number*)

New in version 2.0. Change *diatonic_pitch_class_number* to diatonic pitch-class name:

```
>>> pitchtools.diatonic_pitch_class_number_to_diatonic_pitch_class_name(0)
'c'
```

Return string.

pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_name

`pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_name` (*diatonic_pitch_name*)

New in version 2.0. Change *diatonic_pitch_name* to chromatic pitch-class name:

```
>>> pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_name("c' ' ")
'c'
```

Return string.

pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_number

`pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_number` (*diatonic_pitch_name*)

New in version 2.0. Change *diatonic_pitch_name* to chromatic pitch-class number:

```
>>> pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_number("c' ' ")
0
```

Return integer.

pitchtools.diatonic_pitch_name_to_chromatic_pitch_name

`pitchtools.diatonic_pitch_name_to_chromatic_pitch_name` (*diatonic_pitch_name*)

New in version 2.0. Change *diatonic_pitch_name* to chromatic pitch name:

```
>>> pitchtools.diatonic_pitch_name_to_chromatic_pitch_name("c' ' ")
"c' ' "
```

Return string.

`pitchtools.diatonic_pitch_name_to_chromatic_pitch_number`

`pitchtools.diatonic_pitch_name_to_chromatic_pitch_number` (*diatonic_pitch_name*)

New in version 2.0. Change *diatonic_pitch_name* to chromatic pitch number:

```
>>> pitchtools.diatonic_pitch_name_to_chromatic_pitch_number("c'")
12
```

Return integer.

`pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_name`

`pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_name` (*diatonic_pitch_name*)

New in version 2.0. Change *diatonic_pitch_name* to diatonic pitch-class name:

```
>>> pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_name("c'")
'c'
```

Return string.

`pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_number`

`pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_number` (*diatonic_pitch_name*)

New in version 2.0. Change *diatonic_pitch_name* to diatonic pitch-class number:

```
>>> pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_number("c'")
0
```

Return integer.

`pitchtools.diatonic_pitch_name_to_diatonic_pitch_number`

`pitchtools.diatonic_pitch_name_to_diatonic_pitch_number` (*diatonic_pitch_name*)

New in version 2.0. Change *diatonic_pitch_name* to diatonic pitch number:

```
>>> pitchtools.diatonic_pitch_name_to_diatonic_pitch_number("c'")
7
```

Return integer.

`pitchtools.diatonic_pitch_number_to_chromatic_pitch_number`

`pitchtools.diatonic_pitch_number_to_chromatic_pitch_number` (*diatonic_pitch_number*)

New in version 2.0. Change *diatonic_pitch_number* to chromatic pitch number:

```
>>> pitchtools.diatonic_pitch_number_to_chromatic_pitch_number(7)
12
```

Return integer.

pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_name

`pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_name` (*diatonic_pitch_number*)

New in version 2.0. Change *diatonic_pitch_number* to diatonic pitch-class name:

```
>>> pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_name(7)
'c'
```

Return string.

pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_number

`pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_number` (*diatonic_pitch_number*)

New in version 2.0. Change *diatonic_pitch_number* to diatonic pitch-class number:

```
>>> pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_number(7)
0
```

Return nonnegative integer.

pitchtools.diatonic_pitch_number_to_diatonic_pitch_name

`pitchtools.diatonic_pitch_number_to_diatonic_pitch_name` (*diatonic_pitch_number*)

New in version 2.0. Change *diatonic_pitch_number* to diatonic pitch name:

```
>>> pitchtools.diatonic_pitch_number_to_diatonic_pitch_name(7)
'c' ''
```

Return string.

pitchtools.expr_has_duplicate_named_chromatic_pitch

`pitchtools.expr_has_duplicate_named_chromatic_pitch` (*expr*)

New in version 2.0. True when *expr* has duplicate named chromatic pitch. Otherwise false:

```
>>> chord = Chord([13, 13, 14], (1, 4))
>>> pitchtools.expr_has_duplicate_named_chromatic_pitch(chord)
True
```

Return boolean.

pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class

`pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class` (*expr*)

New in version 2.0. True when *expr* has duplicate numbered chromatic pitch-class. Otherwise false:

```
>>> chord = Chord([1, 13, 14], (1, 4))
>>> pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class(chord)
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.expr_has_duplicate_numeric_chromatic_pitch` to `pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class`().

`pitchtools.expr_to_melodic_chromatic_interval_segment`

`pitchtools.expr_to_melodic_chromatic_interval_segment` (*expr*)

New in version 2.0. Change *expr* to melodic chromatic interval segment:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
>>> pitchtools.expr_to_melodic_chromatic_interval_segment(staff)
MelodicChromaticIntervalSegment(+2, +2, +1, +2, +2, +2, +1)
```

Return melodic chromatic interval segment.

`pitchtools.get_named_chromatic_pitch_from_pitch_carrier`

`pitchtools.get_named_chromatic_pitch_from_pitch_carrier` (*pitch_carrier*)

New in version 1.1. Get named chromatic pitch from *pitch_carrier*:

```
>>> pitch = pitchtools.NamedChromaticPitch('df', 5)
>>> pitch
NamedChromaticPitch("df'")
>>> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(pitch)
NamedChromaticPitch("df'")

>>> note = Note(('df', 5), (1, 4))
>>> note
Note("df''4")
>>> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(note)
NamedChromaticPitch("df'")

>>> note = Note(('df', 5), (1, 4))
>>> note.note_head
NoteHead("df'")
>>> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(note.note_head)
NamedChromaticPitch("df'")

>>> chord = Chord([('df', 5)], (1, 4))
>>> chord
Chord("<df''>4")
>>> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(chord)
NamedChromaticPitch("df'")

>>> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(13)
NamedChromaticPitch("cs'")
```

Raise missing pitch error when *pitch_carrier* carries no pitch.

Raise extra pitch error when *pitch_carrier* carries more than one pitch.

Return named chromatic pitch. Changed in version 2.0: renamed `pitchtools.get_pitch()` to `pitchtools.get_named_chromatic_pitch_from_pitch_carrier()`.

`pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier`

`pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier` (*pitch_carrier*)

New in version 2.0. Get numbered chromatic pitch-class from *pitch_carrier*:

```
>>> note = Note("cs'4")
>>> pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier(note)
NumberedChromaticPitchClass(1)
```

Raise missing pitch error on empty chords.

Raise extra pitch error on many-note chords.

Return numbered chromatic pitch-class.	Changed in version 2.0:	renamed
<code>pitchtools.get_numeric_chromatic_pitch_class_from_pitch_carrier()</code>		to
<code>pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier()</code>		

`pitchtools.harmonic_chromatic_interval_class_number_dictionary`

`pitchtools.harmonic_chromatic_interval_class_number_dictionary(pitches)`

New in version 1.1. Change named chromatic pitches to harmonic chromatic interval-class number dictionary:

```
>>> chord = Chord([0, 2, 11], (1, 4))
>>> vector = pitchtools.harmonic_chromatic_interval_class_number_dictionary(
... chord.written_pitches)
>>> vector
{0: 0, 1: 0, 2: 1, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 1, 10: 0, 11: 1}
```

Return dictionary. Changed in version 2.0: renamed `pitchtools.get_interval_vector()` to `pitchtools.harmonic_chromatic_interval_class_number_dictionary()`.

`pitchtools.insert_and_transpose_nested_subruns_in_chromatic_pitch_class_number_list`

`pitchtools.insert_and_transpose_nested_subruns_in_chromatic_pitch_class_number_list(notes, subrun_indicators)`

New in version 1.1. Insert and transpose nested subruns in *chromatic_pitch_class_number_list* according to *subrun_indicators*:

```
>>> notes = [Note(p, (1, 4)) for p in [0, 2, 7, 9, 5, 11, 4]]
>>> subrun_indicators = [(0, [2, 4]), (4, [3, 1])]
>>> pitchtools.insert_and_transpose_nested_subruns_in_chromatic_pitch_class_number_list(
... notes, subrun_indicators)

>>> t = []
>>> for x in notes:
...     try:
...         t.append(x.written_pitch.chromatic_pitch_number)
...     except AttributeError:
...         t.append([y.written_pitch.chromatic_pitch_number for y in x])

>>> t
[0, [5, 7], 2, [4, 0, 6, 11], 7, 9, 5, [10, 6, 8], 11, [7], 4]
```

Set *subrun_indicators* to a list of zero or more (index, length_list) pairs.

For each (index, length_list) pair in *subrun_indicators* the function will read *index* mod `len(notes)` and insert a subrun of length `length_list[0]` immediately after `notes[index]`, a subrun of length `length_list[1]` immediately after `notes[index+1]`, and, in general, a subrun of length `length_list[i]` immediately after `notes[index+i]`, for `i < length(length_list)`.

New subruns are wrapped with lists. These wrapper lists are designed to allow inspection of the structural changes to *notes* immediately after the function returns. For this reason most calls to this function will be followed by `notes = sequencetools.flatten_sequence(notes):`

```
>>> for note in notes: note
...
Note("c'4")
[Note("f'4"), Note("g'4")]
Note("d'4")
[Note("e'4"), Note("c'4"), Note("fs'4"), Note("b'4")]
Note("g'4")
Note("a'4")
Note("f'4")
[Note("bf'4"), Note("fs'4"), Note("af'4")]
Note("b'4")
[Note("g'4")]
Note("e'4")
```

This function is designed to work on a built-in Python list of notes. This function is **not** designed to work on Abjad voices, staves or other containers because the function currently implements no spanner-handling. That is, this function is designed to be used during precomposition when other, similar abstract pitch transforms may be common.

Return list of integers and / or floats. Changed in version 2.0: renamed `pitchtools.insert_transposed_pc_subruns()` to `pitchtools.insert_and_transpose_nested_subruns()`

`pitchtools.instantiate_pitch_and_interval_test_collection`

`pitchtools.instantiate_pitch_and_interval_test_collection()`

New in version 2.0. Instantiate pitch and interval test collection:

```
>>> for x in pitchtools.instantiate_pitch_and_interval_test_collection(): x
...
HarmonicChromaticInterval(1)
HarmonicChromaticIntervalClass(1)
HarmonicCounterpointInterval(1)
HarmonicCounterpointIntervalClass(1)
HarmonicDiatonicInterval('M2')
HarmonicDiatonicIntervalClass('M2')
InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentDiatonicIntervalClass('M2')
MelodicChromaticInterval(+1)
MelodicChromaticIntervalClass(+1)
MelodicCounterpointInterval(1)
MelodicCounterpointIntervalClass(+1)
MelodicDiatonicInterval('+M2')
MelodicDiatonicIntervalClass('+M2')
NamedChromaticPitch('c')
NamedChromaticPitchClass('c')
NamedDiatonicPitch('c')
NamedDiatonicPitchClass('c')
NumberedChromaticPitch(1)
NumberedChromaticPitchClass(1)
NumberedDiatonicPitch(1)
NumberedDiatonicPitchClass(1)
```

Use to test pitch and interval interface consistency.

Return list.

`pitchtools.inventory_aggregate_subsets`

`pitchtools.inventory_aggregate_subsets()`

New in version 2.0. Inventory aggregate subsets:

```
>>> U_star = pitchtools.inventory_aggregate_subsets()
>>> len(U_star)
4096
>>> for pcset in U_star[:20]:
...     pcset
NumberedChromaticPitchClassSet([])
NumberedChromaticPitchClassSet([0])
NumberedChromaticPitchClassSet([1])
NumberedChromaticPitchClassSet([0, 1])
NumberedChromaticPitchClassSet([2])
NumberedChromaticPitchClassSet([0, 2])
NumberedChromaticPitchClassSet([1, 2])
NumberedChromaticPitchClassSet([0, 1, 2])
NumberedChromaticPitchClassSet([3])
NumberedChromaticPitchClassSet([0, 3])
NumberedChromaticPitchClassSet([1, 3])
NumberedChromaticPitchClassSet([0, 1, 3])
NumberedChromaticPitchClassSet([2, 3])
NumberedChromaticPitchClassSet([0, 2, 3])
NumberedChromaticPitchClassSet([1, 2, 3])
NumberedChromaticPitchClassSet([0, 1, 2, 3])
NumberedChromaticPitchClassSet([4])
NumberedChromaticPitchClassSet([0, 4])
NumberedChromaticPitchClassSet([1, 4])
NumberedChromaticPitchClassSet([0, 1, 4])
```

There are 4096 subsets of the aggregate.

This is U^* in [Morris 1987].

Return list of numbered chromatic pitch-class sets.

`pitchtools.inventory_inversion_equivalent_diatonic_interval_classes`

`pitchtools.inventory_inversion_equivalent_diatonic_interval_classes()`

New in version 2.0. Inventory inversion-equivalent diatonic interval-classes:

```
>>> for dic in pitchtools.inventory_inversion_equivalent_diatonic_interval_classes():
...     dic
...
InversionEquivalentDiatonicIntervalClass('P1')
InversionEquivalentDiatonicIntervalClass('aug1')
InversionEquivalentDiatonicIntervalClass('m2')
InversionEquivalentDiatonicIntervalClass('M2')
InversionEquivalentDiatonicIntervalClass('aug2')
InversionEquivalentDiatonicIntervalClass('dim3')
InversionEquivalentDiatonicIntervalClass('m3')
InversionEquivalentDiatonicIntervalClass('M3')
InversionEquivalentDiatonicIntervalClass('dim4')
```

```
InversionEquivalentDiatonicIntervalClass('P4')
InversionEquivalentDiatonicIntervalClass('aug4')
```

There are 11 inversion-equivalent diatonic interval-classes.

It is an open question as to whether octaves should be included.

Return list of inversion-equivalent diatonic interval-classes.

pitchtools.inversion_equivalent_chromatic_interval_class_number_dictionary

pitchtools.inversion_equivalent_chromatic_interval_class_number_dictionary(*pitches*)

New in version 1.1. Change named chromatic *pitches* to inversion-equivalent chromatic interval-class number dictionary:

```
>>> chord = Chord("<c' d' b''>4")
>>> vector = pitchtools.inversion_equivalent_chromatic_interval_class_number_dictionary(
... chord.written_pitches)
>>> for i in range(7):
...     print '\t%s\t%s' % (i, vector[i])
...
0 0
1 1
2 1
3 1
4 0
5 0
6 0
```

Changed in version 2.0: works with quartertones. Return dictionary.

pitchtools.is_alphabetic_accidental_abbreviation

pitchtools.is_alphabetic_accidental_abbreviation(*expr*)

New in version 2.0. True when *expr* is an alphabetic accidental abbreviation. Otherwise false:

```
>>> pitchtools.is_alphabetic_accidental_abbreviation('tqs')
True
```

The regex `^([s]{1,2}|[f]{1,2}|t?q?[fs])!?$` underlies this predicate.

Return boolean.

pitchtools.is_chromatic_pitch_class_name

pitchtools.is_chromatic_pitch_class_name(*expr*)

New in version 2.0. True when *expr* is a chromatic pitch-class name. Otherwise false:

```
>>> pitchtools.is_chromatic_pitch_class_name('fs')
True
```

The regex `^([a-g,A-G])([s]{1,2}|[f]{1,2}|t?q?[fs]|)!?$` underlies this predicate.

Return boolean.

pitchtools.is_chromatic_pitch_class_name_octave_number_pair

`pitchtools.is_chromatic_pitch_class_name_octave_number_pair(expr)`

New in version 1.1. True when *arg* has the form of a chromatic pitch-class / octave number pair. Otherwise false:

```
>>> pitchtools.is_chromatic_pitch_class_name_octave_number_pair(('cs', 5))
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.is_pair()` to `pitchtools.is_chromatic_pitch_class_name_octave_number_pair()`.

pitchtools.is_chromatic_pitch_class_number

`pitchtools.is_chromatic_pitch_class_number(expr)`

New in version 2.0. True *expr* is a chromatic pitch-class number. Otherwise false:

```
>>> pitchtools.is_chromatic_pitch_class_number(1)
True
```

The chromatic pitch-class numbers are equal to the set `[0, 0.5, ..., 11, 11.5]`.

Return boolean.

pitchtools.is_chromatic_pitch_name

`pitchtools.is_chromatic_pitch_name(expr)`

New in version 2.0. True *expr* is a chromatic pitch name. Otherwise false:

```
>>> pitchtools.is_chromatic_pitch_name('c,')
True
```

The regex `^([a-g,A-G])(([s]{1,2}|[f]{1,2}|t?q?[f,s]|)!)?(, +|' +|)$` underlies this predicate.

Return boolean.

pitchtools.is_chromatic_pitch_number

`pitchtools.is_chromatic_pitch_number(expr)`

New in version 2.0. True *expr* is a chromatic pitch number. Otherwise false:

```
>>> pitchtools.is_chromatic_pitch_number(13)
True
```

The chromatic pitch numbers are equal to the set of all integers in union with the set of all integers plus of minus 0.5.

Return boolean.

pitchtools.is_diatonic_pitch_class_name

`pitchtools.is_diatonic_pitch_class_name(expr)`

New in version 2.0. True when *expr* is a diatonic pitch-class name. Otherwise false:

```
>>> pitchtools.is_diatonic_pitch_class_name('c')
True
```

The regex `^[a-g,A-G]$` underlies this predicate.

Return boolean.

`pitchtools.is_diatonic_pitch_class_number`

`pitchtools.is_diatonic_pitch_class_number`(*expr*)

New in version 2.0. True when *expr* is a diatonic pitch-class number. Otherwise false:

```
>>> pitchtools.is_diatonic_pitch_class_number(0)
True
```

The diatonic pitch-class numbers are equal to the set `[0, 1, 2, 3, 4, 5, 6]`.

Return boolean.

`pitchtools.is_diatonic_pitch_name`

`pitchtools.is_diatonic_pitch_name`(*expr*)

New in version 2.0. True when *expr* is a diatonic pitch name. Otherwise false:

```
>>> pitchtools.is_diatonic_pitch_name("c' ")
True
```

The regex `(^[a-g,A-G])(, +|' +|)$` underlies this predicate.

Return boolean.

`pitchtools.is_diatonic_pitch_number`

`pitchtools.is_diatonic_pitch_number`(*expr*)

New in version 2.0. True when *expr* is a diatonic pitch number. Otherwise false:

```
>>> pitchtools.is_diatonic_pitch_number(7)
True
```

The diatonic pitch numbers are equal to the set of integers.

Return boolean.

`pitchtools.is_diatonic_quality_abbreviation`

`pitchtools.is_diatonic_quality_abbreviation`(*expr*)

New in version 2.0. True when *expr* is a diatonic quality abbreviation. Otherwise false:

```
>>> pitchtools.is_diatonic_quality_abbreviation('aug')
True
```

The regex `^M|m|P|aug|dim$` underlies this predicate.

Return boolean.

pitchtools.is_harmonic_diatonic_interval_abbreviation

`pitchtools.is_harmonic_diatonic_interval_abbreviation(expr)`

New in version 2.0. True when *expr* is a harmonic diatonic interval abbreviation. Otherwise false:

```
>>> pitchtools.is_harmonic_diatonic_interval_abbreviation('M9')
True
```

The regex `^(M|m|P|aug|dim)(\d+)$` underlies this predicate.

Return boolean.

pitchtools.is_melodic_diatonic_interval_abbreviation

`pitchtools.is_melodic_diatonic_interval_abbreviation(expr)`

New in version 2.0. True when *expr* is a melodic diatonic interval abbreviation. Otherwise false:

```
>>> pitchtools.is_melodic_diatonic_interval_abbreviation('+M9')
True
```

The regex `^([+,-]?)(M|m|P|aug|dim)(\d+)$` underlies this predicate.

Return boolean.

pitchtools.is_named_chromatic_pitch_token

`pitchtools.is_named_chromatic_pitch_token(pitch_token)`

New in version 1.1. True when *pitch_token* has the form of an Abjad pitch token. Otherwise false:

```
>>> pitchtools.is_named_chromatic_pitch_token(('c', 4))
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.is_pitch_token()` to `pitchtools.is_named_chromatic_pitch_token()`.

pitchtools.is_octave_tick_string

`pitchtools.is_octave_tick_string(expr)`

New in version 2.0. True when *expr* is an octave tick string. Otherwise false:

```
>>> pitchtools.is_octave_tick_string(',,,')
True
```

The regex `^,+|'|+$` underlies this predicate.

Return boolean.

pitchtools.is_pitch_carrier

`pitchtools.is_pitch_carrier(expr)`

New in version 1.1. True when *expr* is an Abjad pitch, note, note-head of chord instance. Otherwise false:

```
>>> note = Note("c'4")
>>> pitchtools.is_pitch_carrier(note)
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.is_carrier()` to `pitchtools.is_pitch_carrier()`.

`pitchtools.is_pitch_class_octave_number_string`

`pitchtools.is_pitch_class_octave_number_string(expr)`

New in version 2.5. True when *expr* is a pitch-class / octave number string. Otherwise false:

```
>>> pitchtools.is_pitch_class_octave_number_string('C#2')
True
```

Quartertone accidentals are supported.

The regex `^([A-G])([#]{1,2}|[b]{1,2}|[#]?[+]|[b]?[~]|)([-]?[0-9]+)$` underlies this predicate.

Return boolean.

`pitchtools.is_symbolic_accidental_string`

`pitchtools.is_symbolic_accidental_string(expr)`

New in version 2.5. True when *expr* is a symbolic accidental string. Otherwise false:

```
>>> pitchtools.is_symbolic_accidental_string('#+')
True
```

True on empty string.

The regex `^([#]{1,2}|[b]{1,2}|[#]?[+]|[b]?[~]|)$` underlies this predicate.

Return boolean.

`pitchtools.is_symbolic_pitch_range_string`

`pitchtools.is_symbolic_pitch_range_string(expr)`

New in version 2.5. True when *expr* is a symbolic pitch range string. Otherwise false:

```
>>> pitchtools.is_symbolic_pitch_range_string('[A0, C8]')
True
```

The regex that underlies this predicate matches against two comma-separated pitch indicators enclosed in some combination of square brackets and round parentheses.

Return boolean.

`pitchtools.iterate_named_chromatic_pitch_pairs_in_expr`

`pitchtools.iterate_named_chromatic_pitch_pairs_in_expr(expr)`

New in version 2.0. Iterate left-to-right, top-to-bottom named chromatic pitch pairs in *expr*:

```
>>> score = Score([])
>>> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'4")]
>>> score.append(Staff(notes))
>>> notes = [Note(x, (1, 4)) for x in [-12, -15, -17]]
>>> score.append(Staff(notes))
```

```

>>> contexttools.ClefMark('bass')(score[1])
ClefMark('bass')(Staff{3})

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
    g'4
  }
  \new Staff {
    \clef "bass"
    c4
    a,4
    g,4
  }
>>

>>> for pair in pitchtools.iterate_named_chromatic_pitch_pairs_in_expr(score):
...     pair
...
(NamedChromaticPitch("c'"), NamedChromaticPitch('c'))
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"))
(NamedChromaticPitch('c'), NamedChromaticPitch("d'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch('a,'))
(NamedChromaticPitch('c'), NamedChromaticPitch("e'"))
(NamedChromaticPitch('c'), NamedChromaticPitch('a,'))
(NamedChromaticPitch("e'"), NamedChromaticPitch('a,'))
(NamedChromaticPitch("e'"), NamedChromaticPitch("f'"))
(NamedChromaticPitch('a,'), NamedChromaticPitch("f'"))
(NamedChromaticPitch("f'"), NamedChromaticPitch("g'"))
(NamedChromaticPitch("f'"), NamedChromaticPitch('g,'))
(NamedChromaticPitch('a,'), NamedChromaticPitch("g'"))
(NamedChromaticPitch('a,'), NamedChromaticPitch('g,'))
(NamedChromaticPitch("g'"), NamedChromaticPitch('g,'))

```

Chords are handled correctly.

```

>>> chord_1 = Chord([0, 2, 4], (1, 4))
>>> chord_2 = Chord([17, 19], (1, 4))
>>> staff = Staff([chord_1, chord_2])

>>> f(staff)
\new Staff {
  <c' d' e'>4
  <f' g'>4
}

>>> for pair in pitchtools.iterate_named_chromatic_pitch_pairs_in_expr(staff):
...     print pair
...
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("f'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("g'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("f'"))

```

```
(NamedChromaticPitch("d'"), NamedChromaticPitch("g'"))
(NamedChromaticPitch("e'"), NamedChromaticPitch("f'"))
(NamedChromaticPitch("e'"), NamedChromaticPitch("g'"))
(NamedChromaticPitch("f'"), NamedChromaticPitch("g'"))
```

Return generator.

`pitchtools.list_chromatic_pitch_numbers_in_expr`

`pitchtools.list_chromatic_pitch_numbers_in_expr(expr)`

New in version 2.0. List chromatic pitch numbers in *expr*:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> pitchtools.list_chromatic_pitch_numbers_in_expr(tuplet)
(0, 2, 4)
```

Return tuple of zero or more numbers.

`pitchtools.list_harmonic_chromatic_intervals_in_expr`

`pitchtools.list_harmonic_chromatic_intervals_in_expr(expr)`

New in version 2.0. List harmonic chromatic intervals in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> for interval in sorted(pitchtools.list_harmonic_chromatic_intervals_in_expr(staff)):
...     interval
...
HarmonicChromaticInterval(1)
HarmonicChromaticInterval(2)
HarmonicChromaticInterval(2)
HarmonicChromaticInterval(3)
HarmonicChromaticInterval(4)
HarmonicChromaticInterval(5)
```

Return unordered set.

`pitchtools.list_harmonic_diatonic_intervals_in_expr`

`pitchtools.list_harmonic_diatonic_intervals_in_expr(expr)`

New in version 2.0. List harmonic diatonic intervals in *expr*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> for interval in sorted(pitchtools.list_harmonic_diatonic_intervals_in_expr(staff)):
...     interval
...
HarmonicDiatonicInterval('m2')
HarmonicDiatonicInterval('M2')
HarmonicDiatonicInterval('M2')
HarmonicDiatonicInterval('m3')
HarmonicDiatonicInterval('M3')
HarmonicDiatonicInterval('P4')
```

Return unordered set.

pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise

`pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise` (*pitch_carriers*,
wrap=False)

New in version 2.0. List inversion-equivalent chromatic interval-classes pairwise between *pitch_carriers*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
}

>>> result = pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise(
... staff, wrap=False)

>>> for x in result: x
...
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(1)

>>> result = pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise(
... staff, wrap=True)

>>> for x in result: x
...
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentChromaticIntervalClass(0)

>>> notes = staff.leaves
>>> notes = list(reversed(notes))

>>> result = pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise(
... notes, wrap=False)

>>> for x in result: x
...
InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
```

```
InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)

>>> result = pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise(
... notes, wrap=True)

>>> for x in result: x
...
InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(2)
InversionEquivalentChromaticIntervalClass(0)
```

When wrap=False do not return pitch_carriers[-1] - pitch_carriers[0] as last in series.

When wrap=True do return pitch_carriers[-1] - pitch_carriers[0] as last in series.

Return list.

`pitchtools.list_melodic_chromatic_interval_numbers_pairwise`

`pitchtools.list_melodic_chromatic_interval_numbers_pairwise` (*pitch_carriers*,
wrap=False)

New in version 1.1. List melodic chromatic interval numbers pairwise between *pitch_carriers*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
}

>>> pitchtools.list_melodic_chromatic_interval_numbers_pairwise(
... staff)
[2, 2, 1, 2, 2, 2, 1]

>>> pitchtools.list_melodic_chromatic_interval_numbers_pairwise(
... staff, wrap=True)
[2, 2, 1, 2, 2, 2, 1, -12]

>>> notes = [
...     Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"),
...     Note("g'8"), Note("a'8"), Note("b'8"), Note("c''8")]
```

```
>>> notes.reverse()

>>> pitchtools.list_melodic_chromatic_interval_numbers_pairwise(
... notes)
[-1, -2, -2, -2, -1, -2, -2]

>>> pitchtools.list_melodic_chromatic_interval_numbers_pairwise(
... notes, wrap=True)
[-1, -2, -2, -2, -1, -2, -2, 12]
```

When `wrap = False` do not return `pitch_carriers[-1] - pitch_carriers[0]` as last in series.

When `wrap = True` do return `pitch_carriers[-1] - pitch_carriers[0]` as last in series.

Return list. Changed in version 2.0: renamed `pitchtools.get_signed_interval_series()` to `pitchtools.list_melodic_chromatic_interval_numbers_pairwise()`.

`pitchtools.list_named_chromatic_pitches_in_expr`

`pitchtools.list_named_chromatic_pitches_in_expr(expr)`

New in version 2.0. List named chromatic pitches in *expr*:

```
>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> beam_spanner = beamtools.BeamSpanner(staff[:])

>>> for x in pitchtools.list_named_chromatic_pitches_in_expr(beam_spanner):
...     x
...
NamedChromaticPitch("c' ")
NamedChromaticPitch("d' ")
NamedChromaticPitch("e' ")
NamedChromaticPitch("f' ")
```

Return tuple.

`pitchtools.list_numbered_chromatic_pitch_classes_in_expr`

`pitchtools.list_numbered_chromatic_pitch_classes_in_expr(expr)`

New in version 2.0. List numbered chromatic pitch-classes in *expr*:

```
>>> chord = Chord("<cs' ' d' ' ef' '>4")

>>> for x in pitchtools.list_numbered_chromatic_pitch_classes_in_expr(chord):
...     x
...
NumberedChromaticPitchClass(1)
NumberedChromaticPitchClass(2)
NumberedChromaticPitchClass(3)
```

Works with notes, chords, defective chords.

Return tuple or zero or more numbered chromatic pitch-classes. Changed in version 2.0: renamed `pitchtools.list_numeric_chromatic_pitch_classes_in_expr()` to `pitchtools.list_numbered_chromatic_pitch_classes_in_expr()`.

`pitchtools.list_octave_transpositions_of_pitch_carrier_within_pitch_range`

`pitchtools.list_octave_transpositions_of_pitch_carrier_within_pitch_range`(*pitch_carrier*,
pitch_range)

New in version 1.1. List octave transpositions of *pitch_carrier* in *pitch_range*:

```
>>> chord = Chord("<c' d' e'>4")
>>> pitch_range = pitchtools.PitchRange(0, 48)

>>> result = pitchtools.list_octave_transpositions_of_pitch_carrier_within_pitch_range(
...     chord, pitch_range)

>>> for chord in result:
...     chord
...
Chord("<c' d' e'>4")
Chord("<c'' d'' e''>4")
Chord("<c''' d''' e'''>4")
Chord("<c'''' d'''' e''''>4")
```

Return list of newly created *pitch_carrier* objects.

`pitchtools.list_ordered_named_chromatic_pitch_pairs_from_expr_1_to_expr_2`

`pitchtools.list_ordered_named_chromatic_pitch_pairs_from_expr_1_to_expr_2`(*expr_1*,
expr_2)

New in version 2.0. List ordered named chromatic pitch pairs from *expr_1* to *expr_2*:

```
>>> chord_1 = Chord([0, 1, 2], (1, 4))
>>> chord_2 = Chord([3, 4], (1, 4))

>>> for pair in pitchtools.list_ordered_named_chromatic_pitch_pairs_from_expr_1_to_expr_2(
...     chord_1, chord_2):
...     pair
(NamedChromaticPitch("c'"), NamedChromaticPitch("ef'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("cs'"), NamedChromaticPitch("ef'"))
(NamedChromaticPitch("cs'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("ef'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("e'"))
```

Return generator.

`pitchtools.list_unordered_named_chromatic_pitch_pairs_in_expr`

`pitchtools.list_unordered_named_chromatic_pitch_pairs_in_expr`(*expr*)

New in version 2.0. List unordered named chromatic pitch pairs in *expr*:

```
>>> chord = Chord("<c' cs' d' ef'>4")

>>> for pair in pitchtools.list_unordered_named_chromatic_pitch_pairs_in_expr(chord):
...     pair
...
(NamedChromaticPitch("c'"), NamedChromaticPitch("cs'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("ef'"))
```



```
(NamedChromaticPitch("cs'"), NamedChromaticPitch("d'"))
(NamedChromaticPitch("cs'"), NamedChromaticPitch("ef'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("ef'"))
```

Return generator.

`pitchtools.make_n_middle_c_centered_pitches`

`pitchtools.make_n_middle_c_centered_pitches(n)`

New in version 2.0. Make n middle-c centered pitches, where $0 < n$:

```
>>> for p in pitchtools.make_n_middle_c_centered_pitches(5): p
NamedChromaticPitch('f')
NamedChromaticPitch('a')
NamedChromaticPitch("c'")
NamedChromaticPitch("e'")
NamedChromaticPitch("g'")

>>> for p in pitchtools.make_n_middle_c_centered_pitches(4): p
NamedChromaticPitch('g')
NamedChromaticPitch('b')
NamedChromaticPitch("d'")
NamedChromaticPitch("f'")
```

Return list of zero or more named chromatic pitches.

`pitchtools.named_chromatic_pitch_and_clef_to_staff_position_number`

`pitchtools.named_chromatic_pitch_and_clef_to_staff_position_number(pitch, clef)`

New in version 2.0. Change named chromatic *pitch* and *clef* to staff position number:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
>>> clef = contexttools.ClefMark('treble')
>>> for note in staff:
...     written_pitch = note.written_pitch
...     number = pitchtools.named_chromatic_pitch_and_clef_to_staff_position_number(
...         written_pitch, clef)
...     print '%s\t%s' % (written_pitch, number)
c'      -6
d'      -5
e'      -4
f'      -3
g'      -2
a'      -1
b'       0
c''      1
```

Return integer.

`pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches`

`pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches(pitch_tokens)`

New in version 2.0. Change named chromatic *pitch_tokens* to named chromatic pitches:

```
>>> pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches([0, 2, ('ef', 4)])
[NamedChromaticPitch("c'"), NamedChromaticPitch("d'"), NamedChromaticPitch("ef'")]
```

Return list of zero or more named chromatic pitches.

`pitchtools.octave_number_to_octave_tick_string`

`pitchtools.octave_number_to_octave_tick_string(octave_number)`

New in version 2.0. Change *octave_number* to octave tick string:

```
>>> for octave_number in range(-1, 9):
...     tick_string = pitchtools.octave_number_to_octave_tick_string(octave_number)
...     print "%s\t%s" % (octave_number, tick_string)
...
-1  ',,',,
0   ',,',
1   ',,
2   ',
3   '
4   '
5   ',,'
6   ',,',
7   ',,',,
8   ',,',,,'
```

Raise type error on noninteger input.

Return string.

`pitchtools.octave_tick_string_to_octave_number`

`pitchtools.octave_tick_string_to_octave_number(tick_string)`

New in version 2.0. Change *tick_string* to octave number:

```
>>> pitchtools.octave_tick_string_to_octave_number("'")
4
```

Raise type error on nonstring input.

Raise value error on input not of tick string format.

Return integer.

`pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number`

`pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number(pentatonic_scale_degree, transpose=1, phase=0)`

New in version 1.1. Changed *pentatonic_scale_degree* number to chromatic pitch number:

```
>>> for pentatonic_scale_degree in range(9):
...     result = pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number(
...         pentatonic_scale_degree)
...     print '%s %4s' % (pentatonic_scale_degree, result)
...
0  1
```

```

1  3
2  6
3  8
4 10
5 13
6 15
7 18
8 20

```

Pentatonic scale degrees may be negative:

```

>>> for pentatonic_scale_degree in range(-1, -9, -1):
...     result = pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number(
...         pentatonic_scale_degree)
...     print '%s %4s' % (pentatonic_scale_degree, result)
...
-1 -2
-2 -4
-3 -6
-4 -9
-5 -11
-6 -14
-7 -16
-8 -18

```

Return integer. Changed in version 2.0: renamed `pitchtools.pentatonic_to_chromatic()` to `pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number()`.

`pitchtools.permute_named_chromatic_pitch_carrier_list_by_twelve_tone_row`

`pitchtools.permute_named_chromatic_pitch_carrier_list_by_twelve_tone_row` (*pitches*, *row*)

New in version 2.0. Permute named chromatic pitch carrier list by twelve-tone *row*:

```

>>> notes = notetools.make_notes([17, -10, -2, 11], [Duration(1, 4)])
>>> row = pitchtools.TwelveToneRow([10, 0, 2, 6, 8, 7, 5, 3, 1, 9, 4, 11])
>>> pitchtools.permute_named_chromatic_pitch_carrier_list_by_twelve_tone_row(notes, row)
[Note('bf4'), Note('d4'), Note("f'4"), Note("b'4")]

```

Function works by reference only. No objects are copied.

Return list.

`pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name`

`pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name` (*pitch_class_octave_number_string*)

New in version 2.5. Change *pitch_class_octave_number_string* to chromatic pitch name:

```

>>> pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name('C#+2')
'ctqs,'

```

Return string.

`pitchtools.register_chromatic_pitch_class_numbers_by_chromatic_pitch_number_aggregate`

`pitchtools.register_chromatic_pitch_class_numbers_by_chromatic_pitch_number_aggregate` (*pitch_class_numbers*, *aggregates*)

New in version 1.1. Register chromatic *pitch_class_numbers* by chromatic pitch-number *aggregate*:

```
>>> pitchtools.register_chromatic_pitch_class_numbers_by_chromatic_pitch_number_aggregate(
...     [10, 0, 2, 6, 8, 7, 5, 3, 1, 9, 4, 11],
...     [10, 19, 20, 23, 24, 26, 27, 29, 30, 33, 37, 40])
[10, 24, 26, 30, 20, 19, 29, 27, 37, 33, 40, 23]
```

Return list of zero or more chromatic pitch numbers.

`pitchtools.respell_named_chromatic_pitches_in_expr_with_flats`

`pitchtools.respell_named_chromatic_pitches_in_expr_with_flats` (*expr*)

New in version 1.1. Respell named chromatic pitches in *expr* with flats:

```
>>> staff = Staff(notetools.make_repeated_notes(6))
>>> pitchtools.set_ascending_named_chromatic_pitches_on_tie_chains_in_expr(staff)

>>> f(staff)
\new Staff {
    c'8
    cs'8
    d'8
    ef'8
    e'8
    f'8
}

>>> pitchtools.respell_named_chromatic_pitches_in_expr_with_flats(staff)

>>> f(staff)
\new Staff {
    c'8
    df'8
    d'8
    ef'8
    e'8
    f'8
}
```

Return `none`. Changed in version 2.0: renamed `pitchtools.make_flat()` to `pitchtools.respell_named_chromatic_pitches_in_expr_with_flats()`.

`pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps`

`pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps` (*expr*)

New in version 1.1. Respell named chromatic pitches in *expr* with sharps:

```
>>> staff = Staff(notetools.make_repeated_notes(6))
>>> pitchtools.set_ascending_named_chromatic_pitches_on_tie_chains_in_expr(staff)
```

```

>>> f(staff)
\new Staff {
    c'8
    cs'8
    d'8
    ef'8
    e'8
    f'8
}

>>> pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps(staff)

>>> f(staff)
\new Staff {
    c'8
    cs'8
    d'8
    ds'8
    e'8
    f'8
}

```

Return `none`. Changed in version 2.0: renamed `pitchtools.make_sharp()` to `pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps()`.

pitchtools.set_ascending_named_chromatic_pitches_on_tie_chains_in_expr

`pitchtools.set_ascending_named_chromatic_pitches_on_tie_chains_in_expr(expr)`

New in version 1.1. Set ascending named chromatic pitches on nontied pitched components in *expr*:

```

>>> voice = Voice(notetools.make_notes(0, [(5, 32)] * 4))
>>> pitchtools.set_ascending_named_chromatic_pitches_on_tie_chains_in_expr(voice)

>>> f(voice)
\new Voice {
    c'8 ~
    c'32
    cs'8 ~
    cs'32
    d'8 ~
    d'32
    ef'8 ~
    ef'32
}

```

Used primarily in generating test file examples.

Return `none`. Changed in version 2.0: renamed `pitchtools.chromaticize()` to `pitchtools.set_ascending_named_chromatic_pitches_on_tie_chains_in_expr()`.

pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr

`pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(expr,`

key_signature=None)

New in version 1.1. Set ascending named diatonic pitches on nontied pitched components in *expr*:

```
>>> staff = Staff(notetools.make_notes(0, [(5, 32)] * 4))
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(staff)

>>> f(staff)
\new Staff {
  c'8 ~
  c'32
  d'8 ~
  d'32
  e'8 ~
  e'32
  f'8 ~
  f'32
}
```

Used primarily in generating test file examples. New in version 2.0: Optional *key_signature* key-word argument. Return none. Changed in version 2.0: renamed `pitchtools.diatonicize()` to `pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr()`.

`pitchtools.set_default_accidental_spelling`

`pitchtools.set_default_accidental_spelling(spelling='mixed')`

New in version 1.1. Set default accidental spelling to sharps:

```
>>> from abjad.tools import configurationtools

>>> pitchtools.set_default_accidental_spelling('sharps')

>>> [Note(13, (1, 4)), Note(15, (1, 4))]
[Note("cs''4"), Note("ds''4")]
```

Set default accidental spelling to flats:

```
>>> pitchtools.set_default_accidental_spelling('flats')

>>> [Note(13, (1, 4)), Note(15, (1, 4))]
[Note("df''4"), Note("ef''4")]
```

Set default accidental spelling to mixed:

```
>>> pitchtools.set_default_accidental_spelling()

>>> [Note(13, (1, 4)), Note(15, (1, 4))]
[Note("cs''4"), Note("ef''4")]
```

Mixed is system default.

Mixed test case must appear last here for doc tests to check correctly.

Return none. Changed in version 2.0: renamed `pitchtools.change_default_accidental_spelling()` to `pitchtools.set_default_accidental_spelling()`. Changed in version 2.9: renamed `configurationtools.set_default_accidental_spelling()` to `pitchtools.set_default_accidental_spelling()`.

pitchtools.set_written_pitch_of_pitched_components_in_expr

`pitchtools.set_written_pitch_of_pitched_components_in_expr` (*expr*, *written_pitch=0*)

New in version 2.9. Set written pitch of pitched components in *expr* to *written_pitch*:

```
>>> staff = Staff("c' d' e' f'")

>>> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
}

>>> pitchtools.set_written_pitch_of_pitched_components_in_expr(staff)

>>> f(staff)
\new Staff {
    c'4
    c'4
    c'4
    c'4
}
```

Use as a way of neutralizing pitch information in an arbitrary piece of score.

Return none.

pitchtools.sort_named_chromatic_pitch_carriers_in_expr

`pitchtools.sort_named_chromatic_pitch_carriers_in_expr` (*pitch_carriers*)

New in version 2.0. List named chromatic pitch carriers in *expr* sorted by numbered chromatic pitch-class:

```
>>> chord = Chord([9, 11, 12, 14, 16], (1, 4))
>>> notes = chordtools.arpeggiate_chord(chord)

>>> pitchtools.sort_named_chromatic_pitch_carriers_in_expr(notes)
[Note("c''4"), Note("d''4"), Note("e''4"), Note("a'4"), Note("b'4")]
```

The elements in *pitch_carriers* are not changed in any way.

Return list.

pitchtools.spell_chromatic_interval_number

`pitchtools.spell_chromatic_interval_number` (*diatonic_interval_number*, *chromatic_interval_number*)

New in version 2.0. Spell *chromatic_interval_number* according to *diatonic_interval_number*:

```
>>> pitchtools.spell_chromatic_interval_number(2, 1)
MelodicDiatonicInterval('+m2')
```

Return melodic diatonic interval.

pitchtools.spell_chromatic_pitch_number

`pitchtools.spell_chromatic_pitch_number` (*chromatic_pitch_number*, *tonic_pitch_class_name*) *dia-*

New in version 1.1. Spell *chromatic_pitch_number* according to *diatonic_pitch_class_name*:

```
>>> pitchtools.spell_chromatic_pitch_number(14, 'c')
(Accidental('ss'), 5)
```

Return accidental / octave-number pair. Changed in version 2.0: re-named `pitchtools.number_letter_to_accidental_octave()` to `pitchtools.spell_chromatic_pitch_number()`.

pitchtools.split_chromatic_pitch_class_name

`pitchtools.split_chromatic_pitch_class_name` (*chromatic_pitch_class_name*)

New in version 1.1. Change *chromatic_pitch_class_name* to diatonic pitch-class name / alphabetic accidental abbreviation pair:

```
>>> pitchtools.split_chromatic_pitch_class_name('cs')
('c', 's')
```

Return pair of strings. Changed in version 2.0: renamed `pitchtools.name_to_letter_accidental()` to `pitchtools.split_chromatic_pitch_class_name()`.

pitchtools.suggest_clef_for_named_chromatic_pitches

`pitchtools.suggest_clef_for_named_chromatic_pitches` (*pitches*)

New in version 1.1. Suggest clef for named chromatic *pitches*:

```
>>> staff = Staff(notetools.make_notes(range(-12, -6), [(1, 4)]))
>>> pitchtools.suggest_clef_for_named_chromatic_pitches(staff)
ClefMark('bass')
```

Suggest clef based on minimal number of ledger lines.

Return clef mark. Changed in version 2.0: renamed `pitchtools.suggest_clef()` to `pitchtools.suggest_clef_for_named_chromatic_pitches()`.

pitchtools.symbolic_accidental_string_to_alphabetic_accidental_abbreviation

`pitchtools.symbolic_accidental_string_to_alphabetic_accidental_abbreviation` (*symbolic_accidental_string*)

New in version 2.5. Change *symbolic_accidental_string* to alphabetic accidental abbreviation:

```
>>> pitchtools.symbolic_accidental_string_to_alphabetic_accidental_abbreviation('#+')
'tqs'
```

None when *symbolic_accidental_string* is not a valid symbolic accidental string.

Return string or none.

pitchtools.transpose_chromatic_pitch_by_melodic_chromatic_interval_segment

`pitchtools.transpose_chromatic_pitch_by_melodic_chromatic_interval_segment` (*pitch*, *segment*)

New in version 2.0. Transpose chromatic *pitch* by melodic chromatic interval *segment*:

```
>>> ncp = pitchtools.NumberedChromaticPitch(0)
>>> mcis = pitchtools.MelodicChromaticIntervalSegment([0, -1, 2])
>>> pitchtools.transpose_chromatic_pitch_by_melodic_chromatic_interval_segment(ncp, mcis)
[NumberedChromaticPitch(0), NumberedChromaticPitch(-1), NumberedChromaticPitch(1)]
```

Transpose by each interval in *segment* such that each transposition transposes the resulting pitch of the previous transposition.

Return list of numbered chromatic pitches.

pitchtools.transpose_chromatic_pitch_class_number_to_chromatic_pitch_number_neighbor

`pitchtools.transpose_chromatic_pitch_class_number_to_chromatic_pitch_number_neighbor` (*chromatic_pitch_class_number*, *chromatic_pitch_number*)

New in version 1.1. Transpose *chromatic_pitch_class_number* by octaves to nearest neighbor of *chromatic_pitch_number*:

```
>>> pitchtools.transpose_chromatic_pitch_class_number_to_chromatic_pitch_number_neighbor(
...     12, 4)
16
```

Resulting chromatic pitch number must be within one tritone of *chromatic_pitch_number*.

Return chromatic pitch number.

pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping

`pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping` (*chromatic_pitch_number*, *mapping*)

New in version 1.1. Transpose *chromatic_pitch_number* by the some number of octaves up or down. Derive correct number of octaves from *mapping* where *mapping* is a list of (*range_spec*, *octave*) pairs and *range_spec* is, in turn, a (*start*, *stop*) pair suitable to pass to the built-in Python `range()` function:

```
>>> mapping = [((-39, -13), 0), ((-12, 23), 12), ((24, 48), 24)]
```

The mapping given here comprises three (*range_spec*, *octave*) pairs. The first such pair is `((-39, -13), 0)` and can be read as follows: “any pitches between -39 and -13 should be transposed into the octave rooted at pitch 0.” The octave rooted at pitch 0 equals the twelve pitches `range(0, 0 + 12)` or `[0, 1, ..., 10, 11]`.

The second (*range_spec*, *octave*) pair is `((-12, 23), 12)` and can be read as “any pitches between -12 and 23 should be transposed into the octave rooted at pitch 12,” with the octave rooted at pitch 12 equal to the twelve pitches `range(12, 12 + 12)` or `[12, 13, ..., 22, 23]`.

The third and last (*range_spec*, *octave*) pair is `((24, 48), 24)` and can be read as “any pitches between 24 and 48 should be transposed to the octave rooted at 24,” with the octave rooted at 24 equal to the twelve pitches `range(24, 24, + 12)` or `[24, 25, ..., 34, 35]`.

The mapping given here divides the compass of the piano, from -39 to 48 , into three disjunct subranges and then explains how to transpose pitches found in any of those three disjunct subranges. This means that, for example, all the f-sharps within the range of the piano now undergo a known transposition under *mapping* as defined here:

```
>>> pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping(
...     -30, mapping)
6
```

We verify that pitch -30 should map to pitch 6 by noticing that pitch -30 falls in the first of the three subranges defined by *mapping* from -39 to -13 and then noting that *mapping* sends pitches with that subrange to the octave rooted at pitch 0 . The octave transposition of -30 that falls within the octave rooted at 0 is 6 :

```
>>> pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping(
...     -18, mapping)
6
```

Likewise, *mapping* sends pitch -18 to pitch 6 because pitch -18 falls in the same subrange from -39 to -13 as did pitch -39 and so undergoes the same transposition to the octave rooted at 0 .

In this way we can map all f-sharps from -39 to 48 according to *mapping*:

```
>>> pitch_numbers = [-30, -18, -6, 6, 18, 30, 42]
>>> for n in pitch_numbers:
...     n, pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping(
...         n, mapping)
...
(-30, 6)
(-18, 6)
(-6, 18)
(6, 18)
(18, 18)
(30, 30)
(42, 30)
```

And so on.

Return chromatic pitch number. Changed in version 2.0: renamed
`pitchtools.send_pitch_number_to_octave()` to `pitchtools.transpose_chromatic_pitch_number_`

`pitchtools.transpose_named_chromatic_pitch_by_melodic_chromatic_interval_and_respell`

`pitchtools.transpose_named_chromatic_pitch_by_melodic_chromatic_interval_and_respell` (*pitch*,
staff_space,
melodic

New in version 1.1. Transpose named chromatic pitch by *melodic_chromatic_interval* and respell *staff_spaces* above or below:

```
>>> pitch = pitchtools.NamedChromaticPitch(0)

>>> pitchtools.transpose_named_chromatic_pitch_by_melodic_chromatic_interval_and_respell(
...     pitch, 1, 0.5)
NamedChromaticPitch("dtqf' ")
```

Return new named chromatic pitch. Changed in version 2.0: renamed
`pitchtools.staff_space_transpose()` to `pitchtools.transpose_named_chromatic_pitch_by_melo`

pitchtools.transpose_pitch_carrier_by_melodic_interval

`pitchtools.transpose_pitch_carrier_by_melodic_interval` (*pitch_carrier*,
melodic_interval)

New in version 2.0. Transpose *pitch_carrier* by diatonic *melodic_interval*:

```
>>> chord = Chord("<c' e' g'>4")
```

```
>>> pitchtools.transpose_pitch_carrier_by_melodic_interval(chord, '+m2')
Chord("<df' f' af'>4")
```

Transpose *pitch_carrier* by chromatic *melodic_interval*:

```
>>> chord = Chord("<c' e' g'>4")
```

```
>>> pitchtools.transpose_pitch_carrier_by_melodic_interval(chord, 1)
Chord("<cs' f' af'>4")
```

Return non-pitch-carrying input unchanged:

```
>>> rest = Rest('r4')
```

```
>>> pitchtools.transpose_pitch_carrier_by_melodic_interval(rest, 1)
Rest('r4')
```

Return *pitch_carrier*.

pitchtools.transpose_pitch_expr_into_pitch_range

`pitchtools.transpose_pitch_expr_into_pitch_range` (*pitch_expr*, *pitch_range*)

New in version 2.0. Transpose *pitch_expr* into *pitch_range*:

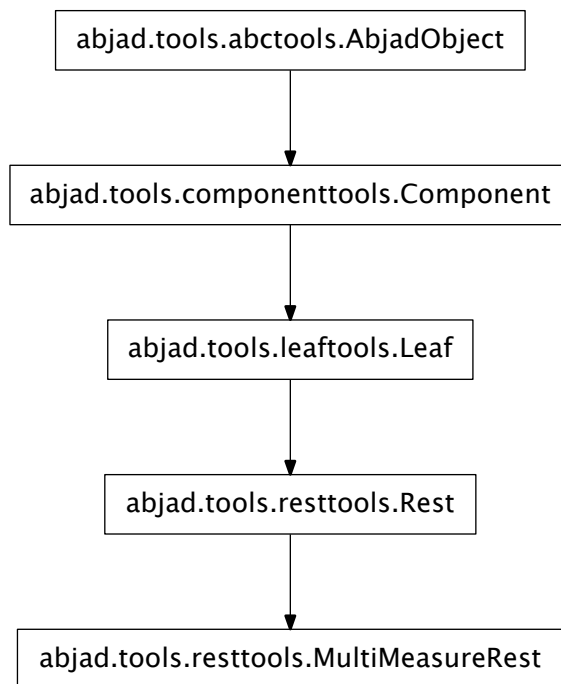
```
>>> pitchtools.transpose_pitch_expr_into_pitch_range(
...     [-2, -1, 13, 14], pitchtools.PitchRange(0, 12))
[10, 11, 1, 2]
```

Return new *pitch_expr* object.

`resttools`

concrete classes

`resttools.MultiMeasureRest`



class `resttools.MultiMeasureRest` (**args, **kwargs*)

New in version 2.0. Abjad model of a multi-measure rest:

```
>>> resttools.MultiMeasureRest((1, 4))
MultiMeasureRest('R4')
```

Multi-measure rests are immutable.

Read-only Properties

`MultiMeasureRest.duration_in_seconds`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.leaf_index`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.lilypond_format`

Inherited from `componenttools.Component`

`MultiMeasureRest.multiplied_duration`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.override`

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

`MultiMeasureRest.parent`

Inherited from `componenttools.Component`

`MultiMeasureRest.preprolated_duration`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.prolated_duration`

Inherited from `componenttools.Component`

`MultiMeasureRest.prolation`

Inherited from `componenttools.Component`

`MultiMeasureRest.set`

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

`MultiMeasureRest.spanners`

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`MultiMeasureRest.start_offset`

Read-only start offset of component.

Inherited from `componenttools.Component`

`MultiMeasureRest.start_offset_in_seconds`

Read-only start offset of comonent in seconds.

Inherited from `componenttools.Component`

`MultiMeasureRest.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`MultiMeasureRest.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`MultiMeasureRest.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`MultiMeasureRest.duration_multiplier`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.written_duration`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.written_pitch_indication_is_at_sounding_pitch`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.written_pitch_indication_is_nonsemantic`
Inherited from `leaftools.Leaf`

Special Methods

`MultiMeasureRest.__and__(arg)`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__mul__(n)`

Inherited from `componenttools.Component`

`MultiMeasureRest.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__or__(arg)`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.__repr__()`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.__rmul__(n)`

Inherited from `componenttools.Component`

`MultiMeasureRest.__str__()`

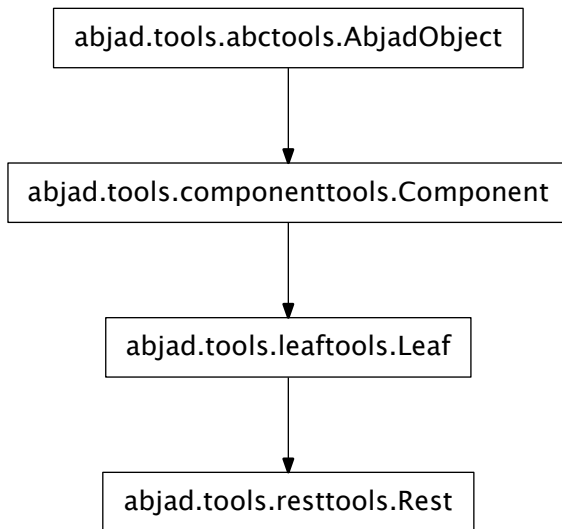
Inherited from `leaftools.Leaf`

`MultiMeasureRest.__sub__(arg)`

Inherited from `leaftools.Leaf`

`MultiMeasureRest.__xor__(arg)`
 Inherited from `leaftools.Leaf`

`resttools.Rest`



```
class resttools.Rest (*args, **kwargs)
    Abjad model of a rest:
```

```
>>> Rest((3, 16))
Rest('r8.')
```

Read-only Properties

`Rest.duration_in_seconds`
 Inherited from `leaftools.Leaf`

`Rest.leaf_index`
 Inherited from `leaftools.Leaf`

`Rest.lilypond_format`
 Inherited from `componenttools.Component`

`Rest.multiplied_duration`
 Inherited from `leaftools.Leaf`

`Rest.override`
 Read-only reference to LilyPond grob override component plug-in.
 Inherited from `componenttools.Component`

`Rest.parent`
 Inherited from `componenttools.Component`

Rest.**preprolated_duration**

Inherited from `leaftools.Leaf`

Rest.**prolated_duration**

Inherited from `componenttools.Component`

Rest.**prolation**

Inherited from `componenttools.Component`

Rest.**.set**

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Rest.**.spanners**

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Rest.**.start_offset**

Read-only start offset of component.

Inherited from `componenttools.Component`

Rest.**.start_offset_in_seconds**

Read-only start offset of comonent in seconds.

Inherited from `componenttools.Component`

Rest.**.stop_offset**

Read-only stop offset of component.

Inherited from `componenttools.Component`

Rest.**.stop_offset_in_seconds**

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Rest.**.storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Rest.**duration_multiplier**

Inherited from `leaftools.Leaf`

Rest.**written_duration**

Inherited from `leaftools.Leaf`

Rest.**written_pitch_indication_is_at_sounding_pitch**

Inherited from `leaftools.Leaf`

Rest.**written_pitch_indication_is_nonsemantic**

Inherited from `leaftools.Leaf`

Special Methods

Rest.**__and__**(*arg*)

Inherited from `leaftools.Leaf`

`Rest.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`Rest.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`Rest.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`Rest.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`Rest.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`Rest.__mul__(n)`
Inherited from `componenttools.Component`

`Rest.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`Rest.__or__(arg)`
Inherited from `leaftools.Leaf`

`Rest.__repr__()`
Inherited from `leaftools.Leaf`

`Rest.__rmul__(n)`
Inherited from `componenttools.Component`

`Rest.__str__()`
Inherited from `leaftools.Leaf`

`Rest.__sub__(arg)`
Inherited from `leaftools.Leaf`

`Rest.__xor__(arg)`
Inherited from `leaftools.Leaf`

functions

resttools.all_are_rests

`resttools.all_are_rests(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad rests:

```
>>> rests = [Rest('r4'), Rest('r4'), Rest('r4')]
```

```
>>> resttools.all_are_rests(rests)
True
```

True when *expr* is an empty sequence:

```
>>> resttools.all_are_rests([])
True
```

Otherwise false:

```
>>> resttools.all_are_rests('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

resttools.is_lilypond_rest_string

`resttools.is_lilypond_rest_string(expr)`

New in version 2.0. True when *expr* is a LilyPond rest string:

```
>>> resttools.is_lilypond_rest_string('r4.. * 1/2')
True
```

Otherwise false:

```
>>> resttools.is_lilypond_rest_string('text')
False
```

The regex `^(r|R)\s*(1|2|4|8|16|32|64|128|\breve|\longa|\maxima)\s*(\.)\s*(*\s*(\d+(\.|\d+` underlies this predicate.

Return boolean.

resttools.make_multi_measure_rests

`resttools.make_multi_measure_rests(duration_tokens)`

New in version 2.0. Make multi-measure rests from *duration_tokens*:

```
>>> resttools.make_multi_measure_rests([(4, 4), (7, 4)])
[MultiMeasureRest('R1'), MultiMeasureRest('R1..')]
```

Return list.

resttools.make_repeated_rests_from_time_signature

`resttools.make_repeated_rests_from_time_signature` (*time_signature*)

New in version 2.0. Make repeated rests from *time_signature*:

```
>>> resttools.make_repeated_rests_from_time_signature((5, 32))
[Rest('r32'), Rest('r32'), Rest('r32'), Rest('r32'), Rest('r32')]
```

Return list of newly constructed rests.

resttools.make_repeated_rests_from_time_signatures

`resttools.make_repeated_rests_from_time_signatures` (*time_signatures*)

Make repeated rests from *time_signatures*:

```
resttools.make_repeated_rests_from_time_signatures([(2, 8), (3, 32)])
[[Rest('r8'), Rest('r8')], [Rest('r32'), Rest('r32'), Rest('r32')]]
```

Return two-dimensional list of newly constructed rest lists.

Use `sequencetools.flatten_sequence()` to flatten output if required.

resttools.make_rests

`resttools.make_rests` (*duration_tokens*, *big_endian=True*, *tied=False*)

New in version 1.1. Make rests.

Make big-endian rests:

```
>>> resttools.make_rests([(5, 16), (9, 16)], big_endian=True)
[Rest('r4'), Rest('r16'), Rest('r2'), Rest('r16')]
```

Make little-endian rests:

```
>>> resttools.make_rests([(5, 16), (9, 16)], big_endian=False)
[Rest('r16'), Rest('r4'), Rest('r16'), Rest('r2')]
```

Make tied rests:

```
>>> voice = Voice(resttools.make_rests([(5, 16), (9, 16)], tied=True))
```

```
>>> f(voice)
\new Voice {
  r4 ~
  r16
  r2 ~
  r16
}
```

Return list of rests. Changed in version 2.0: renamed `construct.rests()` to `resttools.make_rests()`.

resttools.make_tied_rest

`resttools.make_tied_rest` (*duration*, *big_endian=True*, *tied=False*)

Returns a list of rests to fill given duration.

Rests returned are tie-spanned when `tied=True`.

`resttools.replace_leaves_in_expr_with_rests`

`resttools.replace_leaves_in_expr_with_rests(expr)`

New in version 1.1. Replace leaves in *expr* with rests:

```
>>> staff = Staff(Measure((2, 8), "c'8 d'8") * 2)
>>> resttools.replace_leaves_in_expr_with_rests(staff[0])

>>> f(staff)
\new Staff {
  {
    \time 2/8
    r8
    r8
  }
  {
    c'8
    d'8
  }
}
```

Return `none`.

`resttools.set_vertical_positioning_pitch_on_rest`

`resttools.set_vertical_positioning_pitch_on_rest(rest, pitch)`

New in version 2.0. Set vertical positioning *pitch* on *rest*:

```
>>> rest = Rest((1, 4))

>>> resttools.set_vertical_positioning_pitch_on_rest(rest, "d' ")
Rest('r4')

>>> f(rest)
d' '4 \rest
```

Raise type error when *rest* is not a rest.

Return *rest*.

`resttools.yield_groups_of_rests_in_sequence`

`resttools.yield_groups_of_rests_in_sequence(sequence)`

New in version 2.0. Yield groups of rests in *sequence*:

```
>>> staff = Staff("c'8 d'8 r8 r8 <e' g'>8 <f' a'>8 g'8 a'8 r8 r8 <b' d'>8 <c' e'>8")

>>> f(staff)
\new Staff {
  c'8
  d'8
  r8
  r8
```

```

    <e' g'>8
    <f' a'>8
    g'8
    a'8
    r8
    r8
    <b' d''>8
    <c'' e''>8
}

>>> for rest in resttools.yield_groups_of_rests_in_sequence(staff):
...     rest
...
(Rest('r8'), Rest('r8'))
(Rest('r8'), Rest('r8'))

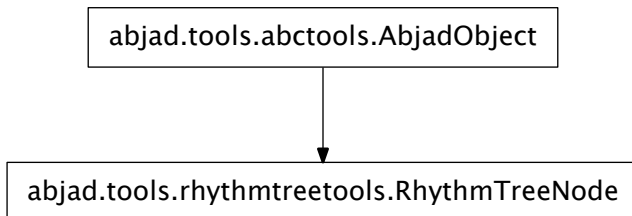
```

Return generator.

rhythmtreetools

abstract classes

rhythmtreetools.RhythmTreeNode



class `rhythmtreetools.RhythmTreeNode` (*duration*)
 Abstract base class of nodes in a rhythm tree structure.

Read-only Properties

`RhythmTreeNode.depth`

The depth of a node in a rhythm-tree structure:

```

>>> rtm = '(4 ((2 (1 1)) (2 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.depth
0

>>> tree[0].depth
1

```

```
>>> tree[0][0].depth
2
```

Return int.

RhythmTreeNode.depthwise_inventory

A dictionary of all nodes in a rhythm-tree, organized by their depth relative the root node:

```
>>> rtm = '(4 ((2 (1 1)) (2 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]
>>> inventory = tree.depthwise_inventory
>>> for depth in sorted(inventory):
...     print 'DEPTH: {}'.format(depth)
...     for node in inventory[depth]:
...         print node.duration, node.offset
...
DEPTH: 0
4 0
DEPTH: 1
2 0
2 2
DEPTH: 2
1 0
1 1
1 2
1 3
```

Return dictionary.

RhythmTreeNode.improper_parentage

The improper parentage of a node in a rhythm-tree, being the sequence of node beginning with itself and ending with the root node of the tree:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.improper_parentage == (a,)
True

>>> b.improper_parentage == (b, a)
True

>>> c.improper_parentage == (c, b, a)
True
```

Return tuple of *RhythmTreeNode* instances.

RhythmTreeNode.offset

The starting offset of a node in a rhythm-tree relative the root:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.offset
Offset(0, 1)
```

```
>>> tree[1].offset
Offset(1, 2)

>>> tree[0][1].offset
Offset(1, 4)
```

Return *Offset* instance.

`RhythmTreeNode.parent`

The node's parent node:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.parent is None
True

>>> b.parent is a
True

>>> c.parent is b
True
```

Return *RhythmTreeNode* instance.

`RhythmTreeNode.parentage_ratios`

A sequence describing the relative durations of the nodes in a node's improper parentage.

The first item in the sequence is the duration of the root node, and subsequent items are pairs of the duration of the next node in the parentage chain and the total duration of that node and its siblings:

```
>>> a = rhythmtreetools.RhythmTreeContainer(1)
>>> b = rhythmtreetools.RhythmTreeContainer(2)
>>> c = rhythmtreetools.RhythmTreeLeaf(3)
>>> d = rhythmtreetools.RhythmTreeLeaf(4)
>>> e = rhythmtreetools.RhythmTreeLeaf(5)

>>> a.extend([b, c])
>>> b.extend([d, e])

>>> a.parentage_ratios
(1,)

>>> b.parentage_ratios
(1, (2, 5))

>>> c.parentage_ratios
(1, (3, 5))

>>> d.parentage_ratios
(1, (2, 5), (4, 9))

>>> e.parentage_ratios
(1, (2, 5), (5, 9))
```

Return tuple.

`RhythmTreeNode.pretty_rtm_format`

The node's pretty-printed RTM format:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]
>>> print tree.pretty_rtm_format
(1 (
  (1 (
    1
    1))
  (1 (
    1
    1)))
```

Return string.

`RhythmTreeNode.prolated_duration`

The prolated duration of the node:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.prolated_duration
Duration(1, 1)

>>> tree[1].prolated_duration
Duration(1, 2)

>>> tree[1][1].prolated_duration
Duration(1, 4)
```

Return *Duration* instance.

`RhythmTreeNode.proper_parentage`

The proper parentage of a node in a rhythm-tree, being the sequence of node beginning with the node's immediate parent and ending with the root node of the tree:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.proper_parentage == ()
True

>>> b.proper_parentage == (a,)
True

>>> c.proper_parentage == (b, a)
True
```

Return tuple of *RhythmTreeNode* instances.

`RhythmTreeNode.root_node`

The root node of the rhythm-tree: that node in the tree which has no parent:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()
```



```
>>> a.append(b)
>>> b.append(c)

>>> a.root_node is a
True
>>> b.root_node is a
True
>>> c.root_node is a
True
```

Return *RhythmTreeNode* instance.

RhythmTreeNode.rtm_format

The node's RTM format:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]
>>> tree.rtm_format
'(1 ((1 (1 1)) (1 (1 1))))'
```

Return string.

RhythmTreeNode.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

RhythmTreeNode.duration

The node's duration in pulses:

```
>>> node = rhythmtreetools.RhythmTreeLeaf(1)
>>> node.duration
1

>>> node.duration = 2
>>> node.duration
2
```

Return int.

Special Methods

RhythmTreeNode.__call__ (*pulse_duration*)

RhythmTreeNode.__eq__ (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

RhythmTreeNode.__ge__ (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

RhythmTreeNode.__getstate__ ()

`RhythmTreeNode.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`RhythmTreeNode.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeNode.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeNode.__ne__(other)`

`RhythmTreeNode.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

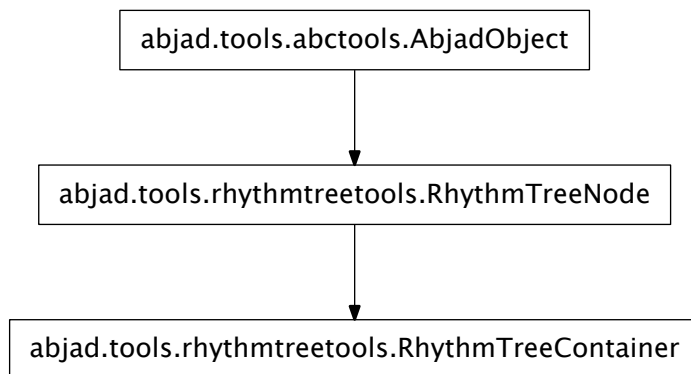
Return string.

Inherited from `abctools.AbjadObject`

`RhythmTreeNode.__setstate__(state)`

concrete classes

`rhythmtreetools.RhythmTreeContainer`



class `rhythmtreetools.RhythmTreeContainer` (*duration=1, children=None*)

A container node in a rhythm tree structure:

```
>>> container = rhythmtreetools.RhythmTreeContainer(duration=1, children=[])
>>> container
RhythmTreeContainer(
    duration=1
)
```

Similar to Abjad containers, *RhythmTreeContainer* supports a list interface, and can be appended, extended, indexed and so forth by other *RhythmTreeNode* subclasses:

```
>>> leaf_a = rhythmtreetools.RhythmTreeLeaf(1)
>>> leaf_b = rhythmtreetools.RhythmTreeLeaf(2)
>>> container.extend([leaf_a, leaf_b])
>>> container
RhythmTreeContainer(
    children=(
        RhythmTreeLeaf(
            duration=1,
            is_pitched=True,
        ),
        RhythmTreeLeaf(
            duration=2,
            is_pitched=True,
        ),
    ),
    duration=1
)

>>> another_container = rhythmtreetools.RhythmTreeContainer(2)
>>> another_container.append(rhythmtreetools.RhythmTreeLeaf(3))
>>> another_container.append(container[1])
>>> container.append(another_container)
>>> container
RhythmTreeContainer(
    children=(
        RhythmTreeLeaf(
            duration=1,
            is_pitched=True,
        ),
        RhythmTreeContainer(
            children=(
                RhythmTreeLeaf(
                    duration=3,
                    is_pitched=True,
                ),
                RhythmTreeLeaf(
                    duration=2,
                    is_pitched=True,
                ),
            ),
            duration=2
        ),
    ),
    duration=1
)
```

Call *RhythmTreeContainer* with a duration to generate a tuplet structure:

```
>>> container((1, 4))
[FixedDurationTuplet(1/4, [c'8, {@ 5:4 c'8., c'8 @])]
```

```
>>> f(_[0])
\times 2/3 {
  c'8
  \times 4/5 {
    c'8.
    c'8
  }
}
```

Returns *RhythmTreeContainer* instance.

Read-only Properties

RhythmTreeContainer.children

The children of a *RhythmTreeContainer* instance:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()
>>> d = rhythmtreetools.RhythmTreeLeaf()
>>> e = rhythmtreetools.RhythmTreeContainer()

>>> a.extend([b, c])
>>> b.extend([d, e])

>>> a.children == (b, c)
True

>>> b.children == (d, e)
True

>>> e.children == ()
True
```

Return tuple of *RhythmTreeNode* instances.

RhythmTreeContainer.contents_duration

The total duration of the children of a *RhythmTreeContainer* instance:

```
>>> rtm = '(1 (1 (2 (1 1 1)) 2))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.contents_duration
5

>>> tree[1].contents_duration
3
```

Return int.

RhythmTreeContainer.depth

The depth of a node in a rhythm-tree structure:

```
>>> rtm = '(4 ((2 (1 1)) (2 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.depth
0
```

```
>>> tree[0].depth
1

>>> tree[0][0].depth
2
```

Return int.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.depthwise_inventory`

A dictionary of all nodes in a rhythm-tree, organized by their depth relative the root node:

```
>>> rtm = '(4 ((2 (1 1)) (2 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]
>>> inventory = tree.depthwise_inventory
>>> for depth in sorted(inventory):
...     print 'DEPTH: {}'.format(depth)
...     for node in inventory[depth]:
...         print node.duration, node.offset
...
DEPTH: 0
4 0
DEPTH: 1
2 0
2 2
DEPTH: 2
1 0
1 1
1 2
1 3
```

Return dictionary.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.improper_parentage`

The improper parentage of a node in a rhythm-tree, being the sequence of node beginning with itself and ending with the root node of the tree:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.improper_parentage == (a,)
True

>>> b.improper_parentage == (b, a)
True

>>> c.improper_parentage == (c, b, a)
True
```

Return tuple of *RhythmTreeNode* instances.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.offset`

The starting offset of a node in a rhythm-tree relative the root:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.offset
Offset(0, 1)

>>> tree[1].offset
Offset(1, 2)

>>> tree[0][1].offset
Offset(1, 4)
```

Return *Offset* instance.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.parent`

The node's parent node:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.parent is None
True

>>> b.parent is a
True

>>> c.parent is b
True
```

Return *RhythmTreeNode* instance.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.parentage_ratios`

A sequence describing the relative durations of the nodes in a node's improper parentage.

The first item in the sequence is the duration of the root node, and subsequent items are pairs of the duration of the next node in the parentage chain and the total duration of that node and its siblings:

```
>>> a = rhythmtreetools.RhythmTreeContainer(1)
>>> b = rhythmtreetools.RhythmTreeContainer(2)
>>> c = rhythmtreetools.RhythmTreeLeaf(3)
>>> d = rhythmtreetools.RhythmTreeLeaf(4)
>>> e = rhythmtreetools.RhythmTreeLeaf(5)

>>> a.extend([b, c])
>>> b.extend([d, e])

>>> a.parentage_ratios
(1,)

>>> b.parentage_ratios
(1, (2, 5))
```

```
>>> c.parentage_ratios
(1, (3, 5))

>>> d.parentage_ratios
(1, (2, 5), (4, 9))

>>> e.parentage_ratios
(1, (2, 5), (5, 9))
```

Return tuple.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.pretty_rtm_format`

The node's pretty-printed RTM format:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]
>>> print tree.pretty_rtm_format
(1 (
  (1 (
    1
    1))
  (1 (
    1
    1))))
```

Return string.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.prolated_duration`

The prolated duration of the node:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.prolated_duration
Duration(1, 1)

>>> tree[1].prolated_duration
Duration(1, 2)

>>> tree[1][1].prolated_duration
Duration(1, 4)
```

Return *Duration* instance.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.proper_parentage`

The proper parentage of a node in a rhythm-tree, being the sequence of node beginning with the node's immediate parent and ending with the root node of the tree:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)
```

```
>>> a.proper_parentage == ()
True

>>> b.proper_parentage == (a,)
True

>>> c.proper_parentage == (b, a)
True
```

Return tuple of *RhythmTreeNode* instances.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.root_node`

The root node of the rhythm-tree: that node in the tree which has no parent:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.root_node is a
True
>>> b.root_node is a
True
>>> c.root_node is a
True
```

Return *RhythmTreeNode* instance.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.rtm_format`

The node's RTM format:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]
>>> tree.rtm_format
'(1 ((1 (1 1)) (1 (1 1))))'
```

Return string.

`RhythmTreeContainer.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`RhythmTreeContainer.duration`

The node's duration in pulses:

```
>>> node = rhythmtreetools.RhythmTreeLeaf(1)
>>> node.duration
1
```



```
>>> node.duration = 2
>>> node.duration
2
```

Return int.

Inherited from `rhythmtreetools.RhythmTreeNode`

Methods

`RhythmTreeContainer.append(node)`

Append *node* to container:

```
>>> a = rhythmtreetools.RhythmTreeContainer(1)
>>> b = rhythmtreetools.RhythmTreeLeaf(2)
>>> c = rhythmtreetools.RhythmTreeLeaf(3)

>>> a
RhythmTreeContainer(
    duration=1
)

>>> a.append(b)
>>> a
RhythmTreeContainer(
    children=(
        RhythmTreeLeaf(
            duration=2,
            is_pitched=True,
        ),
    ),
    duration=1
)

>>> a.append(c)
>>> a
RhythmTreeContainer(
    children=(
        RhythmTreeLeaf(
            duration=2,
            is_pitched=True,
        ),
        RhythmTreeLeaf(
            duration=3,
            is_pitched=True,
        ),
    ),
    duration=1
)
```

It is also possible to append with valid RTM strings, so long as they parse to a single tree:

```
>>> a.append('(7 (1 1 1))')
>>> a
RhythmTreeContainer(
    children=(
        RhythmTreeLeaf(
            duration=2,
            is_pitched=True,

```

```

    ),
    RhythmTreeLeaf(
        duration=3,
        is_pitched=True,
    ),
    RhythmTreeContainer(
        children=(
            RhythmTreeLeaf(
                duration=1,
                is_pitched=True,
            ),
            RhythmTreeLeaf(
                duration=1,
                is_pitched=True,
            ),
            RhythmTreeLeaf(
                duration=1,
                is_pitched=True,
            ),
        ),
        duration=7
    ),
    duration=1
)

```

Return *None*.

`RhythmTreeContainer.extend(expr)`

Extend *expr* against container:

```

>>> a = rhythmtreetools.RhythmTreeContainer(1)
>>> b = rhythmtreetools.RhythmTreeLeaf(2)
>>> c = rhythmtreetools.RhythmTreeLeaf(3)

>>> a
RhythmTreeContainer(
    duration=1
)

>>> a.extend([b, c])
>>> a
RhythmTreeContainer(
    children=(
        RhythmTreeLeaf(
            duration=2,
            is_pitched=True,
        ),
        RhythmTreeLeaf(
            duration=3,
            is_pitched=True,
        ),
    ),
    duration=1
)

```

It is also possible to extend with valid RTM strings:

```

>>> a.extend('(4 (1 1)) (5 (1 1))')
>>> a

```

```
RhythmTreeContainer(
    children=(
        RhythmTreeLeaf(
            duration=2,
            is_pitched=True,
        ),
        RhythmTreeLeaf(
            duration=3,
            is_pitched=True,
        ),
        RhythmTreeContainer(
            children=(
                RhythmTreeLeaf(
                    duration=1,
                    is_pitched=True,
                ),
                RhythmTreeLeaf(
                    duration=1,
                    is_pitched=True,
                ),
            ),
            duration=4
        ),
        RhythmTreeContainer(
            children=(
                RhythmTreeLeaf(
                    duration=1,
                    is_pitched=True,
                ),
                RhythmTreeLeaf(
                    duration=1,
                    is_pitched=True,
                ),
            ),
            duration=5
        ),
    ),
    duration=1
)
```

Return *None*.

`RhythmTreeContainer.index(node)`

Index *node* in container:

```
>>> a = rhythmtreetools.RhythmTreeContainer(1)
>>> b = rhythmtreetools.RhythmTreeLeaf(2)
>>> c = rhythmtreetools.RhythmTreeLeaf(3)

>>> a.extend([b, c])

>>> a.index(b)
0

>>> a.index(c)
1
```

Return nonnegative integer.

`RhythmTreeContainer.insert(i, node)`

Insert *node* in container at index *i*:

```
>>> a = rhythmtreetools.RhythmTreeContainer(1)
>>> b = rhythmtreetools.RhythmTreeLeaf(2)
>>> c = rhythmtreetools.RhythmTreeLeaf(3)
>>> d = rhythmtreetools.RhythmTreeLeaf(100)

>>> a.extend([b, c])

>>> a
RhythmTreeContainer(
  children=(
    RhythmTreeLeaf(
      duration=2,
      is_pitched=True,
    ),
    RhythmTreeLeaf(
      duration=3,
      is_pitched=True,
    ),
  ),
  duration=1
)

>>> a.insert(1, d)

>>> a
RhythmTreeContainer(
  children=(
    RhythmTreeLeaf(
      duration=2,
      is_pitched=True,
    ),
    RhythmTreeLeaf(
      duration=100,
      is_pitched=True,
    ),
    RhythmTreeLeaf(
      duration=3,
      is_pitched=True,
    ),
  ),
  duration=1
)
```

Return *None*.

`RhythmTreeContainer.pop(i=-1)`

Pop node at index *i* from container:

```
>>> a = rhythmtreetools.RhythmTreeContainer(1)
>>> b = rhythmtreetools.RhythmTreeLeaf(2)
>>> c = rhythmtreetools.RhythmTreeLeaf(3)

>>> a.extend([b, c])

>>> a
RhythmTreeContainer(
  children=(
```

```

        RhythmTreeLeaf(
            duration=2,
            is_pitched=True,
        ),
        RhythmTreeLeaf(
            duration=3,
            is_pitched=True,
        ),
    ),
    duration=1
)

>>> node = a.pop()

>>> node == c
True

>>> a
RhythmTreeContainer(
    children=(
        RhythmTreeLeaf(
            duration=2,
            is_pitched=True,
        ),
    ),
    duration=1
)

```

Return node.

`RhythmTreeContainer.remove(node)`

Remove *node* from container:

```

>>> a = rhythmtreetools.RhythmTreeContainer(1)
>>> b = rhythmtreetools.RhythmTreeLeaf(2)
>>> c = rhythmtreetools.RhythmTreeLeaf(3)

>>> a.extend([b, c])

>>> a
RhythmTreeContainer(
    children=(
        RhythmTreeLeaf(
            duration=2,
            is_pitched=True,
        ),
        RhythmTreeLeaf(
            duration=3,
            is_pitched=True,
        ),
    ),
    duration=1
)

>>> a.remove(b)

>>> a
RhythmTreeContainer(
    children=(

```

```

        RhythmTreeLeaf(
            duration=3,
            is_pitched=True,
        ),
    ),
    duration=1
)

```

Return *None*.

Special Methods

`RhythmTreeContainer.__add__(expr)`

Concatenate containers self and expr. The operation `c = a + b` returns a new `RhythmTreeContainer` c with the content of both a and b, and a duration equal to the sum of the durations of a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand:

```

>>> a = rhythmtreetools.RhythmTreeParser() ('(1 (1 1 1))')[0]
>>> b = rhythmtreetools.RhythmTreeParser() ('(2 (3 4))')[0]

```

```

>>> c = a + b

```

```

>>> c.duration
3

```

```

>>> c
RhythmTreeContainer(
  children=(
    RhythmTreeLeaf(
      duration=1,
      is_pitched=True,
    ),
    RhythmTreeLeaf(
      duration=1,
      is_pitched=True,
    ),
    RhythmTreeLeaf(
      duration=1,
      is_pitched=True,
    ),
    RhythmTreeLeaf(
      duration=3,
      is_pitched=True,
    ),
    RhythmTreeLeaf(
      duration=4,
      is_pitched=True,
    ),
  ),
  duration=3
)

```

`RhythmTreeContainer.__call__(pulse_duration)`

Generate Abjad score components:

```

>>> rtm = '(1 (1 (2 (1 1 1)) 2))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

```

```
>>> tree((1, 4))
[FixedDurationTuplet(1/4, [c'16, {@ 3:2 c'16, c'16, c'16 @}, c'8])]
```

Return sequence of components.

`RhythmTreeContainer.__contains__(expr)`

True if `expr` is in container, otherwise False:

```
>>> container = rhythmtreetools.RhythmTreeContainer()
>>> a = rhythmtreetools.RhythmTreeLeaf()
>>> b = rhythmtreetools.RhythmTreeLeaf()
>>> container.append(a)
```

```
>>> a in container
True
```

```
>>> b in container
False
```

Return boolean.

`RhythmTreeContainer.__delitem__(i)`

Find node at index or slice `i` in container and detach from parentage:

```
>>> container = rhythmtreetools.RhythmTreeContainer()
>>> leaf = rhythmtreetools.RhythmTreeLeaf()
```

```
>>> container.append(leaf)
>>> container.children == (leaf,)
True
>>> leaf.parent is container
True
```

```
>>> del(container[0])
```

```
>>> container.children == ()
True
>>> leaf.parent is None
True
```

Return *None*.

`RhythmTreeContainer.__eq__(other)`

True if type, duration and children are equivalent, otherwise False.

Return boolean.

`RhythmTreeContainer.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeContainer.__getitem__(i)`

Return node at index `i` in container:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeLeaf()
>>> c = rhythmtreetools.RhythmTreeContainer()
>>> d = rhythmtreetools.RhythmTreeLeaf()
```

```
>>> e = rhythmtreetools.RhythmTreeLeaf()
>>> f = rhythmtreetools.RhythmTreeLeaf()

>>> a.extend([b, c, f])
>>> c.extend([d, e])

>>> a[0] is b
True
>>> a[1] is c
True
>>> a[2] is f
True
```

Return *RhythmTreeNode* instance.

`RhythmTreeContainer.__getstate__()`
 Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Inherited from `abctools.AbjadObject`

`RhythmTreeContainer.__iter__()`

`RhythmTreeContainer.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeContainer.__len__()`
 Return nonnegative integer number of nodes in container.

`RhythmTreeContainer.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeContainer.__ne__(other)`
 Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeContainer.__repr__()`

`RhythmTreeContainer.__setitem__(i, expr)`
 Set *expr* in self at nonnegative integer index *i*, or set *expr* in self at slice *i*. Replace contents of *self[i]* with *expr*. Attach parentage to contents of *expr*, and detach parentage of any replaced nodes.

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeLeaf()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.parent is a
True
>>> a.children == (b,)
True
```



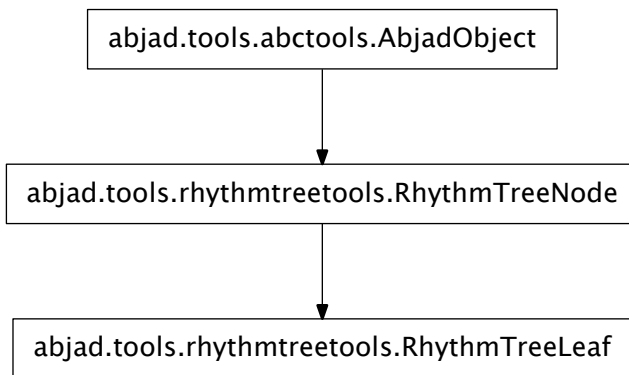
```
>>> a[0] = c

>>> c.parent is a
True
>>> b.parent is None
True
>>> a.children == (c,)
True
```

Return *None*.

RhythmTreeContainer.__setstate__(state)
Inherited from rhythmtreetools.RhythmTreeNode

rhythmtreetools.RhythmTreeLeaf



class rhythmtreetools.**RhythmTreeLeaf**(duration=1, is_pitched=True)

A leaf node in a rhythm tree:

```
>>> leaf = rhythmtreetools.RhythmTreeLeaf(duration=5, is_pitched=True)
>>> leaf
RhythmTreeLeaf(
  duration=5,
  is_pitched=True,
)
```

Call with a pulse duration to generate Abjad leaf objects:

```
>>> result = leaf((1, 8))
>>> result
[Note("c'2"), Note("c'8")]
```

Generates rests when called, if *is_pitched* is False:

```
>>> rhythmtreetools.RhythmTreeLeaf(duration=7, is_pitched=False)((1, 16))
[Rest('r4..')]
```

Return *RhythmTreeLeaf*.

Read-only Properties

`RhythmTreeLeaf.depth`

The depth of a node in a rhythm-tree structure:

```
>>> rtm = '(4 ((2 (1 1)) (2 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.depth
0

>>> tree[0].depth
1

>>> tree[0][0].depth
2
```

Return int.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.depthwise_inventory`

A dictionary of all nodes in a rhythm-tree, organized by their depth relative the root node:

```
>>> rtm = '(4 ((2 (1 1)) (2 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]
>>> inventory = tree.depthwise_inventory
>>> for depth in sorted(inventory):
...     print 'DEPTH: {}'.format(depth)
...     for node in inventory[depth]:
...         print node.duration, node.offset
...
DEPTH: 0
4 0
DEPTH: 1
2 0
2 2
DEPTH: 2
1 0
1 1
1 2
1 3
```

Return dictionary.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.improper_parentage`

The improper parentage of a node in a rhythm-tree, being the sequence of node beginning with itself and ending with the root node of the tree:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.improper_parentage == (a,)
True
```

```
>>> b.improper_parentage == (b, a)
True

>>> c.improper_parentage == (c, b, a)
True
```

Return tuple of *RhythmTreeNode* instances.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.offset`

The starting offset of a node in a rhythm-tree relative the root:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.offset
Offset(0, 1)

>>> tree[1].offset
Offset(1, 2)

>>> tree[0][1].offset
Offset(1, 4)
```

Return Offset instance.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.parent`

The node's parent node:

```
>>> a = rhythmtreetools.RhythmTreeContainer()
>>> b = rhythmtreetools.RhythmTreeContainer()
>>> c = rhythmtreetools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.parent is None
True

>>> b.parent is a
True

>>> c.parent is b
True
```

Return *RhythmTreeNode* instance.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.parentage_ratios`

A sequence describing the relative durations of the nodes in a node's improper parentage.

The first item in the sequence is the duration of the root node, and subsequent items are pairs of the duration of the next node in the parentage chain and the total duration of that node and its siblings:

```
>>> a = rhythmtreetools.RhythmTreeContainer(1)
>>> b = rhythmtreetools.RhythmTreeContainer(2)
>>> c = rhythmtreetools.RhythmTreeLeaf(3)
```

```
>>> d = rhythmtreetools.RhythmTreeLeaf(4)
>>> e = rhythmtreetools.RhythmTreeLeaf(5)

>>> a.extend([b, c])
>>> b.extend([d, e])

>>> a.parentage_ratios
(1,)

>>> b.parentage_ratios
(1, (2, 5))

>>> c.parentage_ratios
(1, (3, 5))

>>> d.parentage_ratios
(1, (2, 5), (4, 9))

>>> e.parentage_ratios
(1, (2, 5), (5, 9))
```

Return tuple.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.pretty_rtm_format`

The node's pretty-printed RTM format:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]
>>> print tree.pretty_rtm_format
(1 (
  (1 (
    1
    1))
  (1 (
    1
    1))))
```

Return string.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.prolated_duration`

The prolated duration of the node:

```
>>> rtm = '(1 ((1 (1 1)) (1 (1 1))))'
>>> tree = rhythmtreetools.RhythmTreeParser()(rtm)[0]

>>> tree.prolated_duration
Duration(1, 1)

>>> tree[1].prolated_duration
Duration(1, 2)

>>> tree[1][1].prolated_duration
Duration(1, 4)
```

Return *Duration* instance.

Inherited from `rhythmtreetools.RhythmTreeNode`

RhythmTreeLeaf.proper_parentage

The proper parentage of a node in a rhythm-tree, being the sequence of node beginning with the node's immediate parent and ending with the root node of the tree:

```
>>> a = rhythmtreertools.RhythmTreeContainer()
>>> b = rhythmtreertools.RhythmTreeContainer()
>>> c = rhythmtreertools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.proper_parentage == ()
True

>>> b.proper_parentage == (a,)
True

>>> c.proper_parentage == (b, a)
True
```

Return tuple of *RhythmTreeNode* instances.

Inherited from `rhythmtreertools.RhythmTreeNode`

RhythmTreeLeaf.root_node

The root node of the rhythm-tree: that node in the tree which has no parent:

```
>>> a = rhythmtreertools.RhythmTreeContainer()
>>> b = rhythmtreertools.RhythmTreeContainer()
>>> c = rhythmtreertools.RhythmTreeLeaf()

>>> a.append(b)
>>> b.append(c)

>>> a.root_node is a
True
>>> b.root_node is a
True
>>> c.root_node is a
True
```

Return *RhythmTreeNode* instance.

Inherited from `rhythmtreertools.RhythmTreeNode`

RhythmTreeLeaf.rtm_format

The node's RTM format:

```
>>> rhythmtreertools.RhythmTreeLeaf(1, is_pitched=True).rtm_format
'1'
>>> rhythmtreertools.RhythmTreeLeaf(5, is_pitched=False).rtm_format
'-5'
```

Return string.

RhythmTreeLeaf.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`RhythmTreeLeaf.duration`

The node's duration in pulses:

```
>>> node = rhythmtreetools.RhythmTreeLeaf(1)
>>> node.duration
1

>>> node.duration = 2
>>> node.duration
2
```

Return int.

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.is_pitched`

True if leaf is pitched:

```
>>> leaf = rhythmtreetools.RhythmTreeLeaf()
>>> leaf.is_pitched
True

>>> leaf.is_pitched = False
>>> leaf.is_pitched
False
```

Return boolean.

Special Methods

`RhythmTreeLeaf.__call__(pulse_duration)`

Generate Abjad score components:

```
>>> leaf = rhythmtreetools.RhythmTreeLeaf(5)
>>> leaf(1, 4)
[Note("c'1"), Note("c'4")]
```

Return sequence of components.

`RhythmTreeLeaf.__eq__(other)`

`RhythmTreeLeaf.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeLeaf.__getstate__()`

Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`RhythmTreeLeaf.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeLeaf.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeLeaf.__ne__(other)`

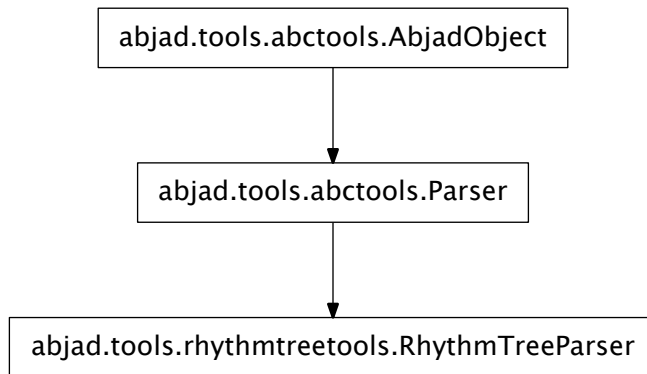
Inherited from `rhythmtreetools.RhythmTreeNode`

`RhythmTreeLeaf.__repr__()`

`RhythmTreeLeaf.__setstate__(state)`

Inherited from `rhythmtreetools.RhythmTreeNode`

`rhythmtreetools.RhythmTreeParser`



class `rhythmtreetools.RhythmTreeParser` (*debug=False*)
 Parses RTM-style rhythm syntax:

```
>>> parser = rhythmtreetools.RhythmTreeParser()
```

```
>>> rtm = '(1 (1 (2 (1 -1 1)) -2))'
```

```
>>> result = parser(rtm)[0]
```

```
>>> result
```

```
RhythmTreeContainer(
  children=(
    RhythmTreeLeaf(
      duration=1,
      is_pitched=True,
    ),
    RhythmTreeContainer(
      children=(
        RhythmTreeLeaf(
          duration=1,
          is_pitched=True,
        ),
        RhythmTreeLeaf(
```

```

        duration=1,
        is_pitched=False,
    ),
    RhythmTreeLeaf(
        duration=1,
        is_pitched=True,
    ),
),
duration=2
),
RhythmTreeLeaf(
    duration=2,
    is_pitched=False,
),
),
duration=1
)

>>> result.rtm_format
'(1 (1 (2 (1 -1 1)) -2))'
```

Returns *RhythmTreeParser* instance.

Read-only Properties

RhythmTreeParser.debug

True if the parser runs in debugging mode.

Inherited from `abctools.Parser`

RhythmTreeParser.lexer

The parser's PLY Lexer instance.

Inherited from `abctools.Parser`

RhythmTreeParser.lexer_rules_object

RhythmTreeParser.logger

The parser's Logger instance.

Inherited from `abctools.Parser`

RhythmTreeParser.logger_path

The output path for the parser's logfile.

Inherited from `abctools.Parser`

RhythmTreeParser.output_path

The output path for files associated with the parser.

Inherited from `abctools.Parser`

RhythmTreeParser.parser

The parser's PLY LRPParser instance.

Inherited from `abctools.Parser`

RhythmTreeParser.parser_rules_object

RhythmTreeParser.pickle_path

The output path for the parser's pickled parsing tables.

Inherited from `abctools.Parser`

`RhythmTreeParser.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`RhythmTreeParser.p_container__LPAREN__INTEGER__node_list_closed__RPAREN`(*p*)
 container : LPAREN INTEGER node_list_closed RPAREN

`RhythmTreeParser.p_error`(*p*)

`RhythmTreeParser.p_leaf__INTEGER`(*p*)
 leaf : INTEGER

`RhythmTreeParser.p_node__container`(*p*)
 node : container

`RhythmTreeParser.p_node__leaf`(*p*)
 node : leaf

`RhythmTreeParser.p_node_list__node_list__node_list_item`(*p*)
 node_list : node_list node_list_item

`RhythmTreeParser.p_node_list__node_list_item`(*p*)
 node_list : node_list_item

`RhythmTreeParser.p_node_list_closed__LPAREN__node_list__RPAREN`(*p*)
 node_list_closed : LPAREN node_list RPAREN

`RhythmTreeParser.p_node_list_item__node`(*p*)
 node_list_item : node

`RhythmTreeParser.p_toplevel__EMPTY`(*p*)
 toplevel :

`RhythmTreeParser.p_toplevel__toplevel__container`(*p*)
 toplevel : toplevel container

`RhythmTreeParser.t_INTEGER`(*t*)
 (-?[1-9]d*)

`RhythmTreeParser.t_error`(*t*)

`RhythmTreeParser.t_newline`(*t*)
 n+

`RhythmTreeParser.tokenize`(*input_string*)
 Tokenize *input_string* and print results.
 Inherited from `abctools.Parser`

Special Methods

`RhythmTreeParser.__call__`(*input_string*)
 Parse *input_string* and return result.

Inherited from `abctools.Parser`

`RhythmTreeParser.__eq__`(*arg*)
 True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`RhythmTreeParser.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeParser.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`RhythmTreeParser.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeParser.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmTreeParser.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`RhythmTreeParser.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

functions

`rhythmtreetools.parse_rtm_syntax`

`rhythmtreetools.parse_rtm_syntax(rtm)`

Parse RTM syntax:

```
>>> from abjad.tools.rhythmtreetools import parse_rtm_syntax
```

```
>>> rtm = '(1 (1 (1 (1 1)) 1))'
```

```
>>> result = parse_rtm_syntax(rtm)
```

```
>>> result
```

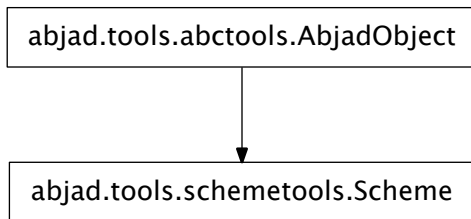
```
FixedDurationTuplet(1/4, [c'8, c'16, c'16, c'8])
```

Return *FixedDurationTuplet* or *Container* instance.

schemetools

concrete classes

schemetools.Scheme



class `schemetools.Scheme` (*args, **kwargs)
 Abjad model of Scheme code:

```
>>> from abjad.tools import schemetools
>>> s = schemetools.Scheme(True)
>>> s.lilypond_format
'##t'
```

`schemetools.Scheme` can represent nested structures:

```
>>> s = schemetools.Scheme(('left', (1, 2, False)), ('right', (1, 2, 3.3)))
>>> s.lilypond_format
'##((left (1 2 #f)) (right (1 2 3.3)))'
```

`schemetools.Scheme` wraps variable-length arguments into a tuple:

```
>>> s = schemetools.Scheme(1, 2, 3)
>>> q = schemetools.Scheme((1, 2, 3))
>>> s.lilypond_format == q.lilypond_format
True
```

`schemetools.Scheme` also takes an optional *quoting* keyword, by which Scheme's various quote, unquote, unquote-splicing characters can be prepended to the formatted result:

```
>>> s = schemetools.Scheme((1, 2, 3), quoting="'#"
>>> s.lilypond_format
"##'(1 2 3)"
```

`schemetools.Scheme` can also force quotes around strings which contain no whitespace:

```
>>> s = schemetools.Scheme('nospaces', force_quotes=True)
>>> f(s)
#"nospaces"
```

The above is useful in certain override situations, as LilyPond's Scheme interpreter will treat unquoted strings as symbols rather than strings.

Scheme is immutable.

Read-only Properties

`Scheme.force_quotes`

`Scheme.lilypond_format`

Hash-mark-prepended format of Scheme:

```
>>> from abjad.tools.schemetools import Scheme
>>> Scheme(True).lilypond_format
'##t'
```

Returns string.

`Scheme.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`Scheme.__eq__` (*other*)

`Scheme.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Scheme.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Scheme.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Scheme.__lt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Scheme.__ne__` (*arg*)

True when `id(self)` does not equal `id(arg)`.

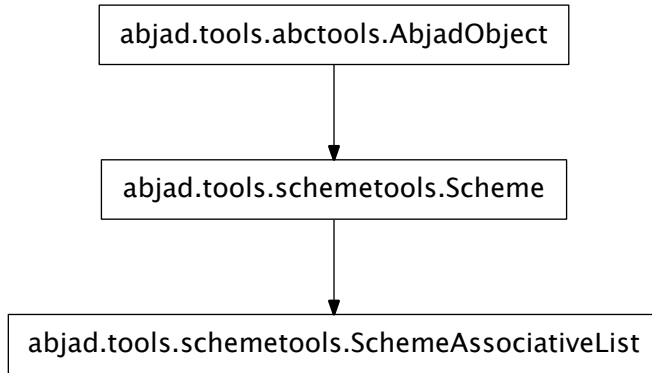
Return boolean.

Inherited from `abctools.AbjadObject`

`Scheme.__repr__` ()

`Scheme.__str__` ()

schemetools.SchemeAssociativeList



class `schemetools.SchemeAssociativeList` (*args, **kwargs)

New in version 2.0. Abjad model of Scheme associative list:

```
>>> from abjad.tools.schemetools import SchemeAssociativeList
>>> SchemeAssociativeList(('space', 2), ('padding', 0.5))
SchemeAssociativeList((SchemePair('space', 2), SchemePair('padding', 0.5)))
```

Scheme associative lists are immutable.

Read-only Properties

`SchemeAssociativeList.force_quotes`

Inherited from `schemetools.Scheme`

`SchemeAssociativeList.lilypond_format`

Hash-mark-prepended format of Scheme:

```
>>> from abjad.tools.schemetools import Scheme
>>> Scheme(True).lilypond_format
'##t'
```

Returns string.

Inherited from `schemetools.Scheme`

`SchemeAssociativeList.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`SchemeAssociativeList.__eq__` (other)

Inherited from `schemetools.Scheme`

`SchemeAssociativeList.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemeAssociativeList.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SchemeAssociativeList.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemeAssociativeList.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemeAssociativeList.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

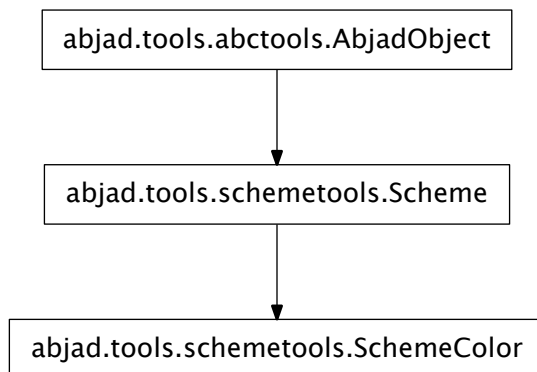
`SchemeAssociativeList.__repr__()`

Inherited from `schemetools.Scheme`

`SchemeAssociativeList.__str__()`

Inherited from `schemetools.Scheme`

`schemetools.SchemeColor`



```
class schemetools.SchemeColor(*args, **kwargs)
```

Abjad model of Scheme color:

```
>>> schemetools.SchemeColor('ForestGreen')
SchemeColor('ForestGreen')
```

Scheme colors are immutable.

Read-only Properties

`SchemeColor.force_quotes`

Inherited from `schemetools.Scheme`

`SchemeColor.lilypond_format`

Hash-mark-prepended format of Scheme:

```
>>> from abjad.tools.schemetools import Scheme
>>> Scheme(True).lilypond_format
'##t'
```

Returns string.

Inherited from `schemetools.Scheme`

`SchemeColor.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`SchemeColor.__eq__(other)`

Inherited from `schemetools.Scheme`

`SchemeColor.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemeColor.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SchemeColor.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemeColor.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

```
SchemeColor.__ne__(arg)
    True when id(self) does not equal id(arg).

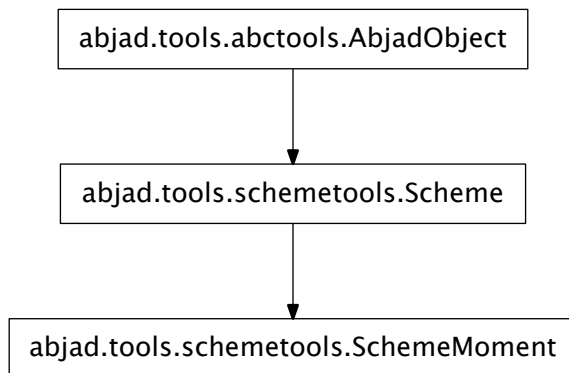
    Return boolean.

    Inherited from abctools.AbjadObject

SchemeColor.__repr__()
    Inherited from schemetools.Scheme

SchemeColor.__str__()
    Inherited from schemetools.Scheme
```

schemetools.SchemeMoment



```
class schemetools.SchemeMoment(*args, **kwargs)
    Abjad model of LilyPond moment:

    >>> schemetools.SchemeMoment(1, 68)
    SchemeMoment((1, 68))
```

Initialize scheme moments with a single fraction, two integers or another scheme moment.

Scheme moments are immutable.

Read-only Properties

`SchemeMoment.duration`

Duration of scheme moment:

```
>>> scheme_moment = schemetools.SchemeMoment(1, 68)
>>> scheme_moment.duration
Duration(1, 68)
```

Return duration.

`SchemeMoment.force_quotes`

Inherited from `schemetools.Scheme`

`SchemeMoment.lilypond_format`

Hash-mark-prepended format of Scheme:

```
>>> from abjad.tools.schemetools import Scheme
>>> Scheme(True).lilypond_format
'##t'
```

Returns string.

Inherited from `schemetools.Scheme`

`SchemeMoment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`SchemeMoment.__eq__(arg)`

`SchemeMoment.__ge__(arg)`

`SchemeMoment.__gt__(arg)`

`SchemeMoment.__le__(arg)`

`SchemeMoment.__lt__(arg)`

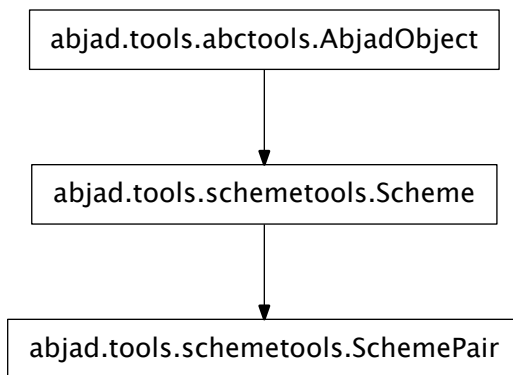
`SchemeMoment.__ne__(arg)`

`SchemeMoment.__repr__()`

`SchemeMoment.__str__()`

Inherited from `schemetools.Scheme`

`schemetools.SchemePair`



class `schemetools.SchemePair(*args, **kwargs)`

Abjad model of Scheme pair:

```
>>> schemetools.SchemePair('spacing', 4)
SchemePair(('spacing', 4))
```

Initialize Scheme pairs with a tuple, two separate values or another Scheme pair.

Scheme pairs are immutable.

Read-only Properties

`SchemePair.force_quotes`

Inherited from `schemetools.Scheme`

`SchemePair.lilypond_format`

`SchemePair.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`SchemePair.__eq__(other)`

Inherited from `schemetools.Scheme`

`SchemePair.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemePair.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SchemePair.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemePair.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemePair.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

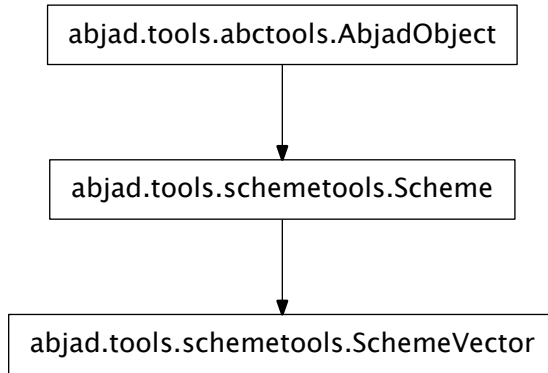
`SchemePair.__repr__()`

Inherited from `schemetools.Scheme`

`SchemePair.__str__()`

Inherited from `schemetools.Scheme`

schemetools.SchemeVector



```

class schemetools.SchemeVector(*args)
    New in version 2.0. Abjad model of Scheme vector:

    >>> schemetools.SchemeVector(True, True, False)
    SchemeVector((True, True, False))
  
```

Scheme vectors and Scheme vector constants differ in only their LilyPond input format.

Scheme vectors are immutable.

Read-only Properties

`SchemeVector.force_quotes`
Inherited from `schemetools.Scheme`

`SchemeVector.lilypond_format`
Hash-mark-prepended format of Scheme:

```

>>> from abjad.tools.schemetools import Scheme
>>> Scheme(True).lilypond_format
'##t'
  
```

Returns string.

Inherited from `schemetools.Scheme`

`SchemeVector.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`SchemeVector.__eq__(other)`
Inherited from `schemetools.Scheme`

`SchemeVector.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemeVector.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SchemeVector.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemeVector.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SchemeVector.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

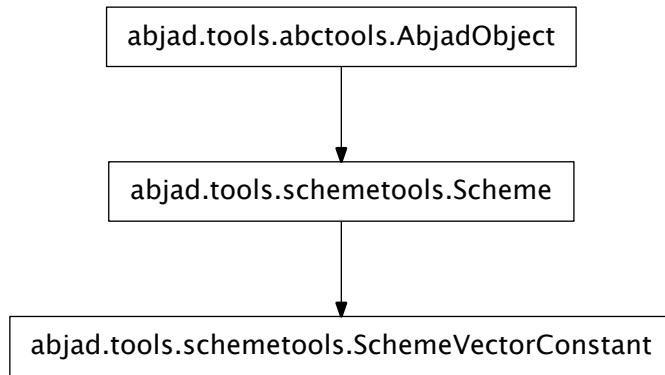
Inherited from `abctools.AbjadObject`

`SchemeVector.__repr__()`

Inherited from `schemetools.Scheme`

`SchemeVector.__str__()`

Inherited from `schemetools.Scheme`

schemetools.SchemeVectorConstant

class `schemetools.SchemeVectorConstant` (*args)
 New in version 2.0. Abjad model of Scheme vector constant:

```
>>> schemetools.SchemeVectorConstant(True, True, False)
SchemeVectorConstant((True, True, False))
```

Scheme vectors and Scheme vector constants differ in only their LilyPond input format.

Scheme vector constants are immutable.

Read-only Properties

`SchemeVectorConstant.force_quotes`

Inherited from `schemetools.Scheme`

`SchemeVectorConstant.lilypond_format`

Hash-mark-prepended format of Scheme:

```
>>> from abjad.tools.schemetools import Scheme
>>> Scheme(True).lilypond_format
'##t'
```

Returns string.

Inherited from `schemetools.Scheme`

`SchemeVectorConstant.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`SchemeVectorConstant.__eq__` (other)

Inherited from `schemetools.Scheme`

`SchemeVectorConstant.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__repr__()`
 Inherited from `schemetools.Scheme`

`SchemeVectorConstant.__str__()`
 Inherited from `schemetools.Scheme`

functions

`schemetools.format_scheme_value`

`schemetools.format_scheme_value(value, force_quotes=False)`
 Format *value* as Scheme would:

```
>>> schemetools.format_scheme_value(1)
'1'

>>> schemetools.format_scheme_value('foo')
'foo'

>>> schemetools.format_scheme_value('bar baz')
'"bar baz"'

>>> schemetools.format_scheme_value([1.5, True, False])
'(1.5 #t #f)'
```

Strings without whitespace can be forcibly quoted via the *force_quotes* keyword:

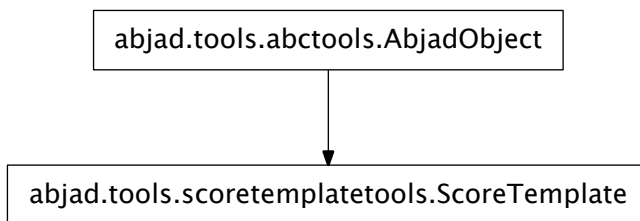
```
>>> schemetools.format_scheme_value('foo', force_quotes=True)
'foo'
```

Return string.

scoretemplatetools

abstract classes

scoretemplatetools.ScoreTemplate



class `scoretemplatetools.ScoreTemplate`
 New in version 2.8. Abstract score template.

Read-only Properties

`ScoreTemplate.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ScoreTemplate.__call__()`

`ScoreTemplate.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ScoreTemplate.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ScoreTemplate.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ScoreTemplate.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ScoreTemplate.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ScoreTemplate.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ScoreTemplate.__repr__()`

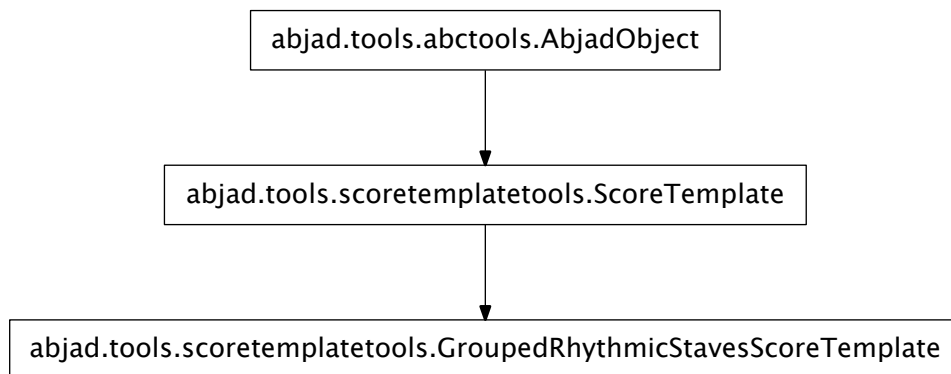
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

concrete classes

`scoretemplatetools.GroupedRhythmicStavesScoreTemplate`



class `scoretemplatetools.GroupedRhythmicStavesScoreTemplate` (*staff_count=2*)

New in version 2.9. Grouped rhythmic staves score template:

```

>>> template = scoretemplatetools.GroupedRhythmicStavesScoreTemplate(staff_count=4)
>>> score = template()
  
```



```

>>> score
Score-"Grouped Rhythmic Staves Score"<<1>>

>>> f(score)
\context Score = "Grouped Rhythmic Staves Score" <<
  \context StaffGroup = "Grouped Rhythmic Staves Staff Group" <<
    \context RhythmicStaff = "Staff 1" {
      \context Voice = "Voice 1" {
      }
    }
    \context RhythmicStaff = "Staff 2" {
      \context Voice = "Voice 2" {
      }
    }
    \context RhythmicStaff = "Staff 3" {
      \context Voice = "Voice 3" {
      }
    }
    \context RhythmicStaff = "Staff 4" {
      \context Voice = "Voice 4" {
      }
    }
  }
>>
>>

```

Return score template object.

Read-only Properties

`GroupedRhythmicStavesScoreTemplate.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`GroupedRhythmicStavesScoreTemplate.__call__()`
`GroupedRhythmicStavesScoreTemplate.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`GroupedRhythmicStavesScoreTemplate.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GroupedRhythmicStavesScoreTemplate.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`GroupedRhythmicStavesScoreTemplate.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GroupedRhythmicStavesScoreTemplate.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GroupedRhythmicStavesScoreTemplate.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

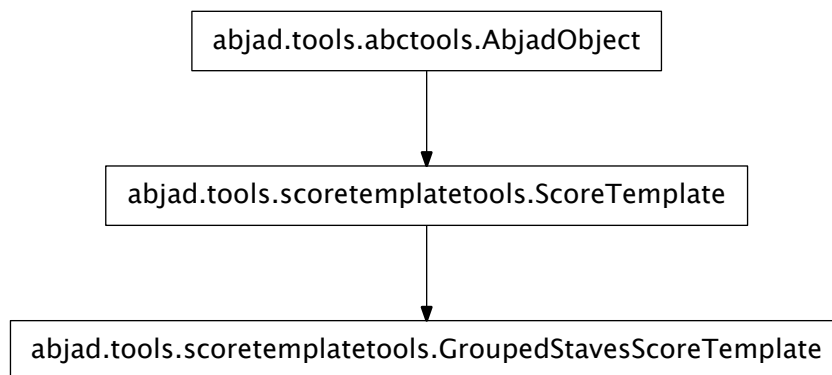
`GroupedRhythmicStavesScoreTemplate.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`scoretemplatetools.GroupedStavesScoreTemplate`



class `scoretemplatetools.GroupedStavesScoreTemplate` (*staff_count*=2)

New in version 2.10. Grouped staves score template:

```
>>> template = scoretemplatetools.GroupedStavesScoreTemplate(staff_count=4)
>>> score = template()
```

```
>>> score
Score-"Grouped Staves Score"<<1>>
```

```
>>> f(score)
\context Score = "Grouped Staves Score" <<
  \context StaffGroup = "Grouped Staves Staff Group" <<
    \context Staff = "Staff 1" {
```

```

        \context Voice = "Voice 1" {
        }
    }
    \context Staff = "Staff 2" {
        \context Voice = "Voice 2" {
        }
    }
    \context Staff = "Staff 3" {
        \context Voice = "Voice 3" {
        }
    }
    \context Staff = "Staff 4" {
        \context Voice = "Voice 4" {
        }
    }
}
>>
>>

```

Return score template object.

Read-only Properties

`GroupedStavesScoreTemplate.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`GroupedStavesScoreTemplate.__call__()`

`GroupedStavesScoreTemplate.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`GroupedStavesScoreTemplate.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GroupedStavesScoreTemplate.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`GroupedStavesScoreTemplate.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GroupedStavesScoreTemplate.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GroupedStavesScoreTemplate.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

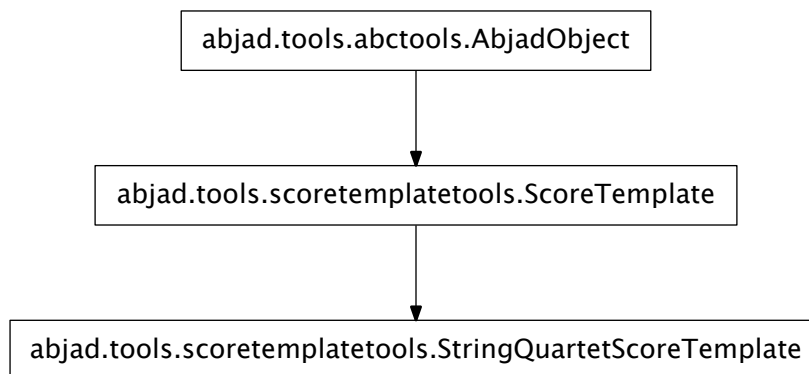
Inherited from `abctools.AbjadObject`

`GroupedStavesScoreTemplate.__repr__()`
 Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`scoretemplatetools.StringQuartetScoreTemplate`



class `scoretemplatetools.StringQuartetScoreTemplate`

New in version 2.8. String quartet score template:

```

>>> template = scoretemplatetools.StringQuartetScoreTemplate()
>>> score = template()

>>> score
Score-"String Quartet Score"<<1>>

>>> f(score)
\context Score = "String Quartet Score" <<
  \context StaffGroup = "String Quartet Staff Group" <<
    \context Staff = "First Violin Staff" {
      \clef "treble"
      \set Staff.instrumentName = \markup { Violin }
      \set Staff.shortInstrumentName = \markup { Vn. }
      \context Voice = "First Violin Voice" {
        }
      }
    }
  }

```

```

\context Staff = "Second Violin Staff" {
  \clef "treble"
  \set Staff.instrumentName = \markup { Violin }
  \set Staff.shortInstrumentName = \markup { Vn. }
  \context Voice = "Second Violin Voice" {
  }
}
\context Staff = "Viola Staff" {
  \clef "alto"
  \set Staff.instrumentName = \markup { Viola }
  \set Staff.shortInstrumentName = \markup { Va. }
  \context Voice = "Viola Voice" {
  }
}
\context Staff = "Cello Staff" {
  \clef "bass"
  \set Staff.instrumentName = \markup { Cello }
  \set Staff.shortInstrumentName = \markup { Vc. }
  \context Voice = "Cello Voice" {
  }
}
>>
>>

```

Return score template.

Read-only Properties

`StringQuartetScoreTemplate.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`StringQuartetScoreTemplate.__call__()`

`StringQuartetScoreTemplate.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`StringQuartetScoreTemplate.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`StringQuartetScoreTemplate.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`StringQuartetScoreTemplate.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`StringQuartetScoreTemplate.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`StringQuartetScoreTemplate.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

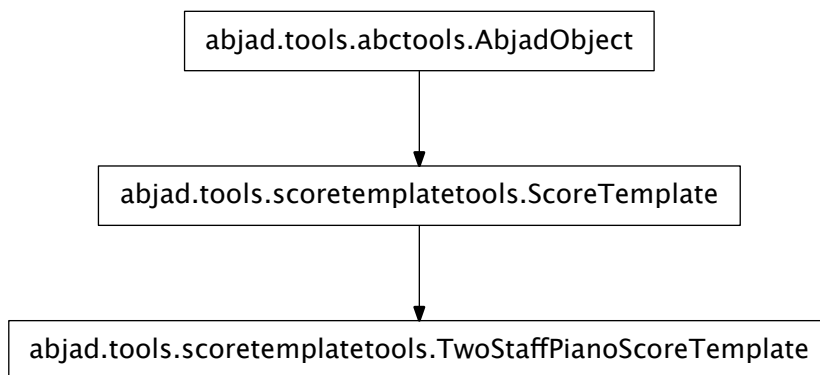
`StringQuartetScoreTemplate.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`scoretemplatetools.TwoStaffPianoScoreTemplate`



class `scoretemplatetools.TwoStaffPianoScoreTemplate`

New in version 2.8. Two-staff piano score template:

```
>>> template = scoretemplatetools.TwoStaffPianoScoreTemplate()
>>> score = template()
```

```
>>> score
Score-"Two-Staff Piano Score"<<1>>
```

```
>>> f(score)
\context Score = "Two-Staff Piano Score" <<
  \context PianoStaff = "Piano Staff" <<
    \set PianoStaff.instrumentName = \markup { Piano }
```

```

        \set PianoStaff.shortInstrumentName = \markup { Pf. }
        \context Staff = "RH Staff" {
            \clef "treble"
            \context Voice = "RH Voice" {
            }
        }
        \context Staff = "LH Staff" {
            \clef "bass"
            \context Voice = "LH Voice" {
            }
        }
    }
>>
>>

```

Return score template.

Read-only Properties

`TwoStaffPianoScoreTemplate.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`TwoStaffPianoScoreTemplate.__call__()`

`TwoStaffPianoScoreTemplate.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

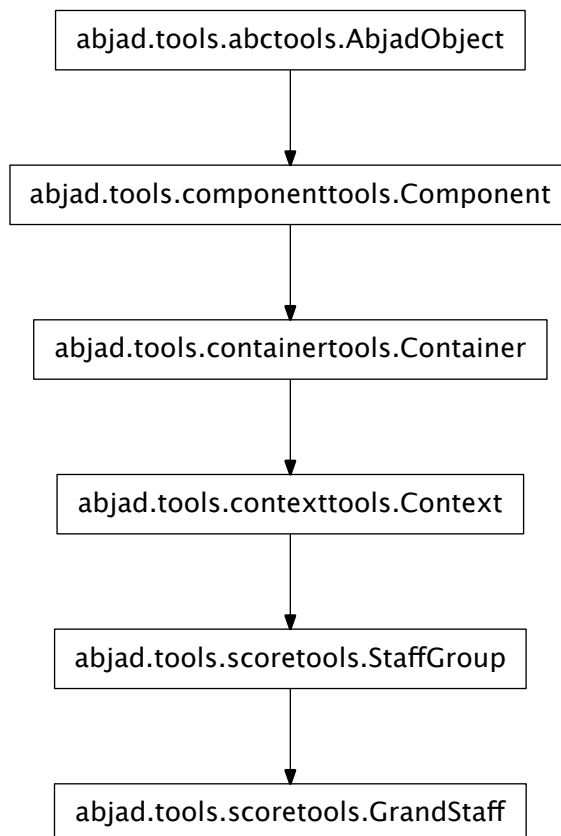
Return string.

Inherited from `abctools.AbjadObject`

`scoretools`

concrete classes

`scoretools.GrandStaff`



class scoretools.**GrandStaff** (*music*, *context_name*='GrandStaff', *name*=None)

Abjad model of grand staff:

```
>>> staff_1 = Staff("c'4 d'4 e'4 f'4 g'1")
>>> staff_2 = Staff("g2 f2 e1")

>>> grand_staff = scoretools.GrandStaff([staff_1, staff_2])

>>> f(grand_staff)
\new GrandStaff <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
    g'1
  }
  \new Staff {
    g2
    f2
    e1
  }
>>
```

Return grand staff.

Read-only Properties

GrandStaff.contents_duration

Inherited from `containertools.Container`

GrandStaff.duration_in_seconds

Inherited from `containertools.Container`

GrandStaff.engraver_consists

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_consists.append('Horizontal_bracket_engraver')
>>> f(staff)
\new Staff \with {
  \consists Horizontal_bracket_engraver
} {
}
```

Inherited from `contexttools.Context`

GrandStaff.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_removals.append('Time_signature_engraver')
>>> f(staff)
\new Staff \with {
  \remove Time_signature_engraver
} {
}
```

Inherited from `contexttools.Context`

GrandStaff.is_semantic

Inherited from `contexttools.Context`

GrandStaff.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

GrandStaff.lilypond_format

Inherited from `contexttools.Context`

GrandStaff.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

GrandStaff.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

GrandStaff.parent

Inherited from `componenttools.Component`

GrandStaff.preprolated_duration

Inherited from `containertools.Container`

GrandStaff.prolated_duration

Inherited from `componenttools.Component`

GrandStaff.prolation

Inherited from `componenttools.Component`

GrandStaff.set

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

GrandStaff.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

GrandStaff.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

`GrandStaff.start_offset_in_seconds`

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

`GrandStaff.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`GrandStaff.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`GrandStaff.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`GrandStaff.context_name`

Read / write name of context as a string.

Inherited from `contexttools.Context`

`GrandStaff.is_nonsemantic`

Set indicator of nonsemantic voice:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (5, 16), (5, 16)])
>>> voice = Voice(measures)
>>> voice.name = 'HiddenTimeSignatureVoice'
```

```
>>> voice.is_nonsemantic = True
```

```
>>> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    s1 * 5/16
  }
}
```

```
>>> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
>>> voice = Voice([])
```

```
>>> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Inherited from `contexttools.Context`

GrandStaff.is_parallel

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')] )

>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}

>>> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True

>>> f(container)
<<
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
>>
```

Return none.

Inherited from `containertools.Container`

GrandStaff.name

Read-write name of context. Must be string or none.

Inherited from `contexttools.Context`

Methods

`GrandStaff.append(component)`

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

`GrandStaff.extend(expr)`

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: *expr* may now be a LilyPond input string. Inherited from `containertools.Container`

`GrandStaff.index(component)`

Index *component* in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> note = container[-1]
>>> note
Note("e'8")
```

```
>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

`GrandStaff.insert(i, component)`

Insert *component* in container at index *i*:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
>>> container.insert(1, Note("cs'8"))
```

```
>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Inherited from `containertools.Container`

`GrandStaff.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
>>> container.pop(-1)
Note("e'8")
```

```
>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

`GrandStaff.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

`GrandStaff.__add__(expr)`

Concatenate containers self and expr. The operation $c = a + b$ returns a new Container c with the content of both a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

`GrandStaff.__contains__(expr)`

True if expr is in container, otherwise False.

Inherited from `containertools.Container`

`GrandStaff.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

`GrandStaff.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`GrandStaff.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GrandStaff.__getitem__(i)`

Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`GrandStaff.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`GrandStaff.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`GrandStaff.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

`GrandStaff.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GrandStaff.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`GrandStaff.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GrandStaff.__mul__(n)`

Inherited from `componenttools.Component`

`GrandStaff.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`GrandStaff.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`GrandStaff.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Inherited from `contexttools.Context`

`GrandStaff.__rmul__(n)`

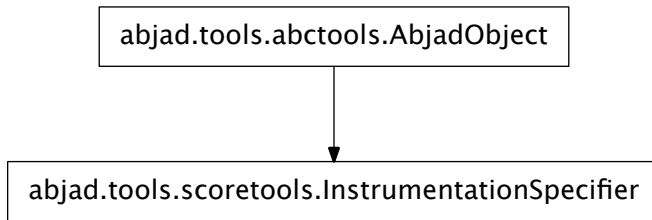
Inherited from `componenttools.Component`

`GrandStaff.__setitem__(i, expr)`

Set ‘expr’ in self at nonnegative integer index i. Or, set ‘expr’ in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with ‘expr’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

`scoretools.InstrumentationSpecifier`



class `scoretools.InstrumentationSpecifier` (*performers=None*)

New in version 2.5. Abjad model of score instrumentation:

```

>>> flute = scoretools.Performer('Flute')
>>> flute.instruments.append(instrumenttools.Flute())
>>> flute.instruments.append(instrumenttools.AltoFlute())

>>> guitar = scoretools.Performer('Guitar')
>>> guitar.instruments.append(instrumenttools.Guitar())

>>> instrumentation_specifier = scoretools.InstrumentationSpecifier([flute, guitar])

>>> z(instrumentation_specifier)
scoretools.InstrumentationSpecifier(
  performers=scoretools.PerformerInventory([
    scoretools.Performer(
      name='Flute',
      instruments=instrumenttools.InstrumentInventory([
        instrumenttools.Flute(),
        instrumenttools.AltoFlute()
      ])
    ),
    scoretools.Performer(
      name='Guitar',
      instruments=instrumenttools.InstrumentInventory([
        instrumenttools.Guitar()
      ])
    )
  ])
)

```

Return instrumentation specifier.

Read-only Properties

`InstrumentationSpecifier.instrument_count`

Read-only number of instruments in score:

```
>>> instrumentation_specifier.instrument_count
3
```

Return nonnegative integer.

`InstrumentationSpecifier.instruments`

Read-only list of instruments derived from performers:

```
>>> instrumentation_specifier.instruments
[Flute(), AltoFlute(), Guitar()]
```

Return list.

`InstrumentationSpecifier.performer_count`

Read-only number of performers in score:

```
>>> instrumentation_specifier.performer_count
2
```

Return nonnegative integer.

`InstrumentationSpecifier.performer_name_string`

Read-only string of performer names:

```
>>> instrumentation_specifier.performer_name_string
'Flute, Guitar'
```

Return string.

`InstrumentationSpecifier.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`InstrumentationSpecifier.performers`

Read / write list of performers in score:

```
>>> z(instrumentation_specifier.performers)
scoretools.PerformerInventory([
    scoretools.Performer(
        name='Flute',
        instruments=instrumenttools.InstrumentInventory([
            instrumenttools.Flute(),
            instrumenttools.AltoFlute()
        ])
    ),
    scoretools.Performer(
        name='Guitar',
        instruments=instrumenttools.InstrumentInventory([
            instrumenttools.Guitar()
        ])
    )
])
```

Return performer inventory.

Special Methods

`InstrumentationSpecifier.__eq__(other)`

`InstrumentationSpecifier.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InstrumentationSpecifier.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`InstrumentationSpecifier.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InstrumentationSpecifier.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InstrumentationSpecifier.__ne__(other)`

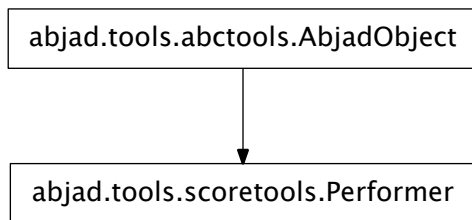
`InstrumentationSpecifier.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

scoretools.Performer



class `scoretools.Performer` (*name=None, instruments=None*)

New in version 2.5. Abjad model of performer:

```
>>> scoretools.Performer(name='flutist')
Performer(name='flutist', instruments=InstrumentInventory([]))
```

The purpose of the class is to model things like flute I doubling piccolo and alto flute.

At present the class is a list of instruments.

Read-only Properties

Performer.instrument_count

Read-only number of instruments to be played by performer:

```
>>> performer = scoretools.Performer('flutist')

>>> performer.instruments.append(instrumenttools.Flute())
>>> performer.instruments.append(instrumenttools.Piccolo())

>>> performer.instrument_count
2
```

Return nonnegative integer

Performer.is_doubling

Is performer to play more than one instrument?

```
>>> performer = scoretools.Performer('flutist')

>>> performer.instruments.append(instrumenttools.Flute())
>>> performer.instruments.append(instrumenttools.Piccolo())

>>> performer.is_doubling
True
```

Return boolean.

Performer.likely_instruments_based_on_performer_name

New in version 2.5. Likely instruments based on performer name:

```
>>> flutist = scoretools.Performer(name='flutist')
>>> for likely_instrument in flutist.likely_instruments_based_on_performer_name:
...     likely_instrument
...
<class 'abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute'>
<class 'abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute'>
<class 'abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute'>
<class 'abjad.tools.instrumenttools.Flute.Flute.Flute'>
<class 'abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo'>
```

Return list.

Performer.most_likely_instrument_based_on_performer_name

New in version 2.5. Most likely instrument based on performer name:

```
>>> flutist = scoretools.Performer(name='flutist')
>>> flutist.most_likely_instrument_based_on_performer_name
<class 'abjad.tools.instrumenttools.Flute.Flute.Flute'>
```

Return instrument class.

Performer.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties**Performer.instruments**

List of instruments to be played by performer:

```
>>> performer = scoretools.Performer('flutist')

>>> performer.instruments.append(instrumenttools.Flute())
>>> performer.instruments.append(instrumenttools.Piccolo())

>>> performer.instruments
InstrumentInventory([Flute(), Piccolo()])
```

Return list.

Performer.name

Score name of performer:

```
>>> performer = scoretools.Performer('flutist')

>>> performer.name
'flutist'
```

Return string.

Methods**Performer.make_performer_name_instrument_dictionary** (*locale=None*)

New in version 2.5. Make performer name / instrument dictionary.

Return dictionary.

Special Methods**Performer.__eq__** (*other*)**Performer.__ge__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`**Performer.__gt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`**Performer.__hash__** ()**Performer.__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`**Performer.__lt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`**Performer.__ne__** (*other*)

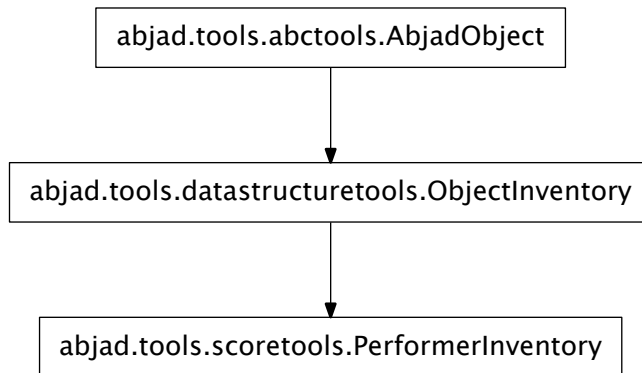
`Performer.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`scoretools.PerformerInventory`



class `scoretools.PerformerInventory` (*tokens=None, name=None*)

New in version 2.8. Abjad model of an ordered list of performers.

Performer inventories implement the list interface and are mutable.

Read-only Properties

`PerformerInventory.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`PerformerInventory.name`

Read / write name of inventory.

Inherited from `datastructuretools.ObjectInventory`

Methods

`PerformerInventory.append(token)`

Change *token* to item and append.

Inherited from `datastructuretools.ObjectInventory`

`PerformerInventory.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

PerformerInventory.**extend**(*tokens*)

Change *tokens* to items and extend.

Inherited from `datastructuretools.ObjectInventory`

PerformerInventory.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

PerformerInventory.**insert**()

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

PerformerInventory.**pop**([*index*]) → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

PerformerInventory.**remove**()

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

PerformerInventory.**reverse**()

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

PerformerInventory.**sort**()

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

PerformerInventory.**__add__**()

`x.__add__(y) <==> x+y`

Inherited from `__builtin__.list`

PerformerInventory.**__contains__**(*token*)

Inherited from `datastructuretools.ObjectInventory`

PerformerInventory.**__delitem__**()

`x.__delitem__(y) <==> del x[y]`

Inherited from `__builtin__.list`

PerformerInventory.**__delslice__**()

`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

PerformerInventory.**__eq__**()

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.list`

PerformerInventory.**__ge__**()

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.list`

```

PerformerInventory.__getitem__ ()
    x.__getitem__(y) <==> x[y]

    Inherited from __builtin__.list

PerformerInventory.__getslice__ ()
    x.__getslice__(i, j) <==> x[i:j]

    Use of negative indices is not supported.

    Inherited from __builtin__.list

PerformerInventory.__gt__ ()
    x.__gt__(y) <==> x>y

    Inherited from __builtin__.list

PerformerInventory.__iadd__ ()
    x.__iadd__(y) <==> x+=y

    Inherited from __builtin__.list

PerformerInventory.__imul__ ()
    x.__imul__(y) <==> x*=y

    Inherited from __builtin__.list

PerformerInventory.__iter__ () <==> iter(x)
    Inherited from __builtin__.list

PerformerInventory.__le__ ()
    x.__le__(y) <==> x<=y

    Inherited from __builtin__.list

PerformerInventory.__len__ () <==> len(x)
    Inherited from __builtin__.list

PerformerInventory.__lt__ ()
    x.__lt__(y) <==> x<y

    Inherited from __builtin__.list

PerformerInventory.__mul__ ()
    x.__mul__(n) <==> x*n

    Inherited from __builtin__.list

PerformerInventory.__ne__ ()
    x.__ne__(y) <==> x!=y

    Inherited from __builtin__.list

PerformerInventory.__repr__ ()
    Inherited from datastructuretools.ObjectInventory

PerformerInventory.__reversed__ ()
    L.__reversed__() – return a reverse iterator over the list

    Inherited from __builtin__.list

PerformerInventory.__rmul__ ()
    x.__rmul__(n) <==> n*x

    Inherited from __builtin__.list

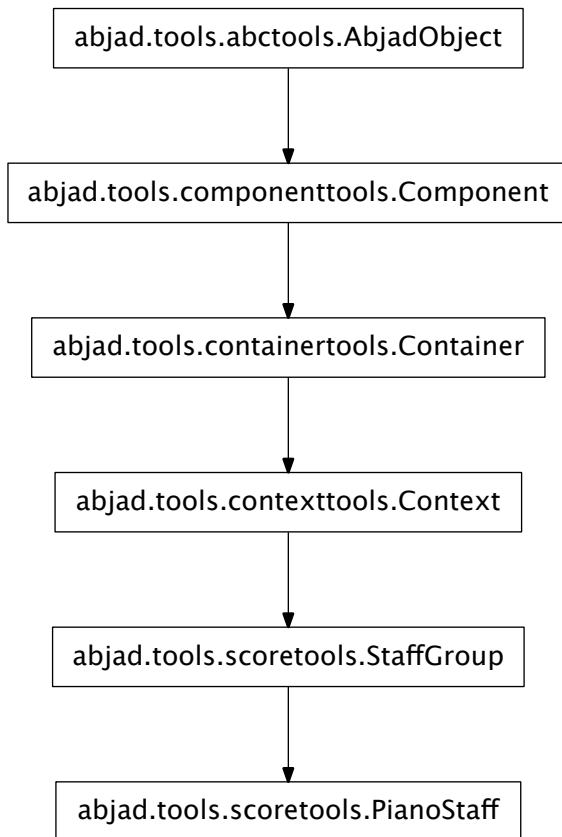
```



```
PerformerInventory.__setitem__ ()
    x.__setitem__(i, y) <==> x[i]=y
    Inherited from __builtin__.list

PerformerInventory.__setslice__ ()
    x.__setslice__(i, j, y) <==> x[i:j]=y
    Use of negative indices is not supported.
    Inherited from __builtin__.list
```

scoretools.PianoStaff



```
class scoretools.PianoStaff (music=None, context_name='PianoStaff', name=None)
    Abjad model of piano staff:

    >>> staff_1 = Staff("c'4 d'4 e'4 f'4 g'1")
    >>> staff_2 = Staff("g2 f2 e1")

    >>> piano_staff = scoretools.PianoStaff([staff_1, staff_2])
```

```
>>> f(piano_staff)
\new PianoStaff <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
    g'1
  }
  \new Staff {
    g2
    f2
    e1
  }
>>
```

Return piano staff.

Read-only Properties

PianoStaff.contents_duration

Inherited from `containertools.Container`

PianoStaff.duration_in_seconds

Inherited from `containertools.Container`

PianoStaff.engraver_consists

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_consists.append('Horizontal_bracket_engraver')
>>> f(staff)
\new Staff \with {
  \consists Horizontal_bracket_engraver
} {
}
```

Inherited from `contexttools.Context`

PianoStaff.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_removals.append('Time_signature_engraver')
>>> f(staff)
\new Staff \with {
  \remove Time_signature_engraver
} {
}
```

Inherited from `contexttools.Context`

PianoStaff.is_semantic

Inherited from `contexttools.Context`

PianoStaff.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

PianoStaff.lilypond_format

Inherited from `contexttools.Context`

PianoStaff.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

PianoStaff.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

PianoStaff.parent

Inherited from `componenttools.Component`

PianoStaff.preprolated_duration

Inherited from `containertools.Container`

PianoStaff.prolated_duration

Inherited from `componenttools.Component`

PianoStaff.prolation

Inherited from `componenttools.Component`

PianoStaff.set

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

PianoStaff.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

PianoStaff.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

PianoStaff.start_offset_in_seconds

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

PianoStaff.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

PianoStaff.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

PianoStaff.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

PianoStaff.context_name

Read / write name of context as a string.

Inherited from `contexttools.Context`

PianoStaff.is_nonsemantic

Set indicator of nonsemantic voice:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (5, 16), (5, 16)])
>>> voice = Voice(measures)
>>> voice.name = 'HiddenTimeSignatureVoice'
```

```
>>> voice.is_nonsemantic = True
```

```
>>> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    s1 * 5/16
  }
}
```

```
>>> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
>>> voice = Voice([])
```

```
>>> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice interaction and other functions.

Inherited from `contexttools.Context`

PianoStaff.is_parallel

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')] )

>>> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

>>> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True

>>> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Inherited from `containertools.Container`

PianoStaff.name

Read-write name of context. Must be string or none.

Inherited from `contexttools.Context`

Methods**PianoStaff.append(*component*)**

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
>>> container.append(Note("f'8"))
```

```
>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

`PianoStaff.extend(expr)`

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])
```

```
>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: *expr* may now be a LilyPond input string. Inherited from `containertools.Container`

`PianoStaff.index(component)`

Index *component* in container:

```
>>> container = Container("c'8 d'8 e'8")
```

```
>>> note = container[-1]
>>> note
Note("e'8")
```

```
>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

`PianoStaff.insert(i, component)`

Insert *component* in container at index *i*:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}

```

Return none.

Inherited from `containertools.Container`

`PianoStaff.pop(i=-1)`

Pop component at index *i* from container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}

```

Return component.

Inherited from `containertools.Container`

`PianoStaff.remove(component)`

Remove *component* from container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8

```

```

        e'8 ]
    }

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

`PianoStaff.__add__(expr)`

Concatenate containers self and expr. The operation `c = a + b` returns a new `Container` c with the content of both a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

`PianoStaff.__contains__(expr)`

True if expr is in container, otherwise False.

Inherited from `containertools.Container`

`PianoStaff.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

`PianoStaff.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`PianoStaff.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PianoStaff.__getitem__(i)`

Return component at index i in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`PianoStaff.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`PianoStaff.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`PianoStaff.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

`PianoStaff.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PianoStaff.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`PianoStaff.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PianoStaff.__mul__(n)`

Inherited from `componenttools.Component`

`PianoStaff.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`PianoStaff.__radd__(expr)`

Extend container by contents of *expr* to the right.

Inherited from `containertools.Container`

`PianoStaff.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Inherited from `contexttools.Context`

`PianoStaff.__rmul__(n)`

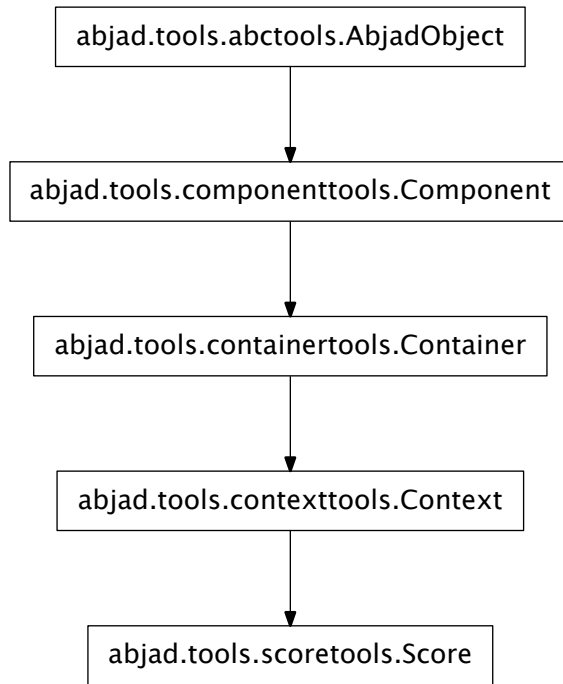
Inherited from `componenttools.Component`

`PianoStaff.__setitem__(i, expr)`

Set ‘*expr*’ in self at nonnegative integer index *i*. Or, set ‘*expr*’ in self at slice *i*. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with ‘*expr*’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

scoretools.Score



class scoretools.**Score** (*music=None, context_name='Score', name=None*)
 Abjad model of a score:

```

>>> staff_1 = Staff("c'8 d'8 e'8 f'8")
>>> staff_2 = Staff("c'8 d'8 e'8 f'8")
>>> score = Score([staff_1, staff_2])
>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>
  
```

Return score object.

Read-only Properties

Score.contents_duration

Inherited from `containertools.Container`

Score.duration_in_seconds

Inherited from `containertools.Container`

Score.engraver_consists

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_consists.append('Horizontal_bracket_engraver')
>>> f(staff)
\new Staff \with {
    \consists Horizontal_bracket_engraver
} {
}
```

Inherited from `contexttools.Context`

Score.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_removals.append('Time_signature_engraver')
>>> f(staff)
\new Staff \with {
    \remove Time_signature_engraver
} {
}
```

Inherited from `contexttools.Context`

Score.is_semantic

Inherited from `contexttools.Context`

Score.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

Score.lilypond_format

Inherited from `contexttools.Context`

Score.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

Score.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Score.parent

Inherited from `componenttools.Component`

Score.preprolated_duration

Inherited from `containertools.Container`

Score.prolated_duration

Inherited from `componenttools.Component`

Score.prolation

Inherited from `componenttools.Component`

Score.set

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Score.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Score.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

Score.start_offset_in_seconds

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

Score.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

Score.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Score.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Score.context_name

Read / write name of context as a string.

Inherited from `contexttools.Context`

Score.is_nonsemantic

Set indicator of nonsemantic voice:

```

>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (5, 16), (5, 16)])
>>> voice = Voice(measures)
>>> voice.name = 'HiddenTimeSignatureVoice'

>>> voice.is_nonsemantic = True

>>> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    s1 * 5/16
  }
}

>>> voice.is_nonsemantic
True

```

Get indicator of nonsemantic voice:

```

>>> voice = Voice([])

>>> voice.is_nonsemantic
False

```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Inherited from `contexttools.Context`

Score.**`is_parallel`**

Get parallel container:

```

>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}

>>> container.is_parallel
False

```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True

>>> f(container)
<<
    \new Voice {
      c'8
      d'8
      e'8
    }
    \new Voice {
      g4.
    }
>>
```

Return none.

Inherited from `containertools.Container`

`Score.name`

Read-write name of context. Must be string or none.

Inherited from `contexttools.Context`

Methods

`Score.append(component)`

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
  f'8
}
```

Return none.

Inherited from `containertools.Container`

`Score.extend(expr)`

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)
```

```

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```

Return none. New in version 2.3: `expr` may now be a LilyPond input string. Inherited from `containertools.Container`

Score.index (*component*)

Index *component* in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2

```

Return nonnegative integer.

Inherited from `containertools.Container`

Score.insert (*i*, *component*)

Insert *component* in container at index *i*:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}

```

Return none.

Inherited from `containertools.Container`

`Score.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

`Score.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

Score.**__add__** (*expr*)

Concatenate containers self and expr. The operation $c = a + b$ returns a new Container *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

Score.**__contains__** (*expr*)

True if *expr* is in container, otherwise False.

Inherited from `containertools.Container`

Score.**__delitem__** (*i*)

Find component(s) at index or slice ‘*i*’ in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

Score.**__eq__** (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Score.**__ge__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Score.**__getitem__** (*i*)

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

Score.**__gt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Score.**__iadd__** (*expr*)

__iadd__ avoids unnecessary copying of structures.

Inherited from `containertools.Container`

Score.**__imul__** (*total*)

Multiply contents of container ‘*total*’ times. Return multiplied container.

Inherited from `containertools.Container`

Score.**__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Score.**__len__** ()

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

Score.**__lt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Score.**__mul__**(*n*)

Inherited from `componenttools.Component`

Score.**__ne__**(*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Score.**__radd__**(*expr*)

Extend container by contents of *expr* to the right.

Inherited from `containertools.Container`

Score.**__repr__**()

Changed in version 2.0. Named contexts now print name at the interpreter.

Inherited from `contexttools.Context`

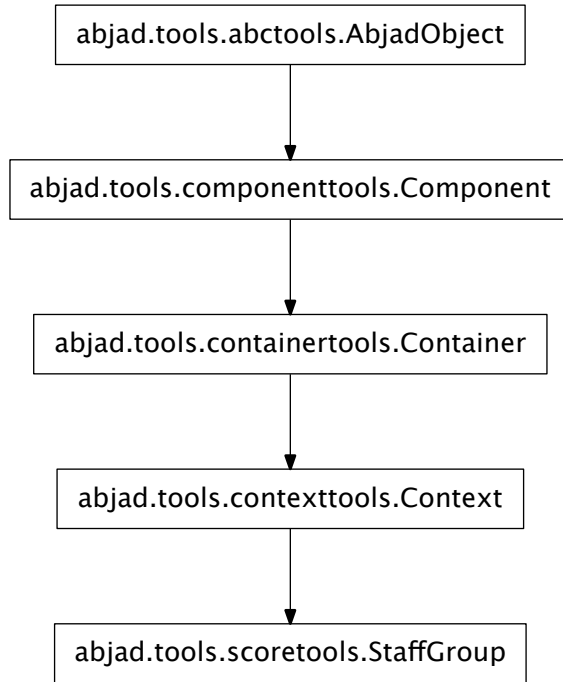
Score.**__rmul__**(*n*)

Inherited from `componenttools.Component`

Score.**__setitem__**(*i, expr*)

Set 'expr' in self at nonnegative integer index *i*. Or, set 'expr' in self at slice *i*. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

scoretools.StaffGroup

class `scoretools.StaffGroup` (*music=None, context_name='StaffGroup', name=None*)
 Abjad model of staff group:

```

>>> staff_1 = Staff("c'4 d'4 e'4 f'4 g'1")
>>> staff_2 = Staff("g2 f2 e1")

>>> staff_group = scoretools.StaffGroup([staff_1, staff_2])

>>> f(staff_group)
\new StaffGroup <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
    g'1
  }
  \new Staff {
    g2
    f2
    e1
  }
>>
  
```

Return staff group.

Read-only Properties

StaffGroup.contents_duration

Inherited from `containertools.Container`

StaffGroup.duration_in_seconds

Inherited from `containertools.Container`

StaffGroup.engraver_consists

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_consists.append('Horizontal_bracket_engraver')
>>> f(staff)
\new Staff \with {
    \consists Horizontal_bracket_engraver
} {
}
```

Inherited from `contexttools.Context`

StaffGroup.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_removals.append('Time_signature_engraver')
>>> f(staff)
\new Staff \with {
    \remove Time_signature_engraver
} {
}
```

Inherited from `contexttools.Context`

StaffGroup.is_semantic

Inherited from `contexttools.Context`

StaffGroup.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

StaffGroup.lilypond_format

Inherited from `contexttools.Context`

StaffGroup.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

`StaffGroup.override`

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

`StaffGroup.parent`

Inherited from `componenttools.Component`

`StaffGroup.preprolated_duration`

Inherited from `containertools.Container`

`StaffGroup.prolated_duration`

Inherited from `componenttools.Component`

`StaffGroup.prolation`

Inherited from `componenttools.Component`

`StaffGroup.set`

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

`StaffGroup.spanners`

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`StaffGroup.start_offset`

Read-only start offset of component.

Inherited from `componenttools.Component`

`StaffGroup.start_offset_in_seconds`

Read-only start offset of comonent in seconds.

Inherited from `componenttools.Component`

`StaffGroup.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`StaffGroup.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`StaffGroup.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`StaffGroup.context_name`

Read / write name of context as a string.

Inherited from `contexttools.Context`

`StaffGroup.is_nonsemantic`

Set indicator of nonsemantic voice:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (5, 16), (5, 16)])
>>> voice = Voice(measures)
>>> voice.name = 'HiddenTimeSignatureVoice'

>>> voice.is_nonsemantic = True

>>> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    s1 * 5/16
  }
}

>>> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
>>> voice = Voice([])

>>> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Inherited from `contexttools.Context`

StaffGroup.is_parallel

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}

>>> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True

>>> f(container)
<<
    \new Voice {
      c'8
      d'8
      e'8
    }
    \new Voice {
      g4.
    }
>>
```

Return none.

Inherited from `containertools.Container`

`StaffGroup.name`

Read-write name of context. Must be string or none.

Inherited from `contexttools.Context`

Methods

`StaffGroup.append(component)`

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
  c'8 [
  d'8
  e'8 ]
  f'8
}
```

Return none.

Inherited from `containertools.Container`

`StaffGroup.extend(expr)`

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: `expr` may now be a LilyPond input string. Inherited from `containertools.Container`

`StaffGroup.index(component)`

Index *component* in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

`StaffGroup.insert(i, component)`

Insert *component* in container at index *i*:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```


Return none.

Inherited from `containertools.Container`

`StaffGroup.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

`StaffGroup.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

`StaffGroup.__add__(expr)`

Concatenate containers self and expr. The operation $c = a + b$ returns a new Container c with the content of both a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

`StaffGroup.__contains__(expr)`

True if expr is in container, otherwise False.

Inherited from `containertools.Container`

`StaffGroup.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

`StaffGroup.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`StaffGroup.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`StaffGroup.__getitem__(i)`

Return component at index i in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`StaffGroup.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`StaffGroup.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`StaffGroup.__imul__(total)`

Multiply contents of container 'total' times. Return multiplied container.

Inherited from `containertools.Container`

`StaffGroup.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`StaffGroup.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`StaffGroup.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`StaffGroup.__mul__(n)`

Inherited from `componenttools.Component`

`StaffGroup.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`StaffGroup.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`StaffGroup.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Inherited from `contexttools.Context`

`StaffGroup.__rmul__(n)`

Inherited from `componenttools.Component`

`StaffGroup.__setitem__(i, expr)`

Set ‘`expr`’ in `self` at nonnegative integer index `i`. Or, set ‘`expr`’ in `self` at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with ‘`expr`’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

functions

`scoretools.add_double_bar_to_end_of_score`

`scoretools.add_double_bar_to_end_of_score(score)`

New in version 2.0. Add double bar to end of `score`:

```
>>> staff = Staff("c'4 d'4 e'4 f'4")

>>> scoretools.add_double_bar_to_end_of_score(staff)
BarLine('|.')(f'4)

>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
  \bar "|."
}
```

Return double bar.

scoretools.add_markup_to_end_of_score

`scoretools.add_markup_to_end_of_score(score, markup, extra_offset=None)`

New in version 2.0. Add *markup* to end of *score*:

```
>>> staff = Staff("c'4 d'4 e'4 f'4")
>>> markup = r'\italic \right-column { "Bremen - Boston - LA." "Jul 2010 - May 2011." }'
>>> markup = markuptools.Markup(markup, Down)
>>> markup = scoretools.add_markup_to_end_of_score(staff, markup, (4, -2))

>>> z(markup)
markuptools.Markup(
  (MarkupCommand('italic', MarkupCommand('right-column',
    ['Bremen - Boston - LA.', 'Jul 2010 - May 2011.']))),
  direction=Down
)

>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  \once \override TextScript #'extra-offset = #'(4 . -2)
  f'4 _ \markup {
    \italic
      \right-column
        {
          "Bremen - Boston - LA."
          "Jul 2010 - May 2011."
        }
  }
}
```

Return *markup*.

scoretools.all_are_scores

`scoretools.all_are_scores(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad scores:

```
>>> score = Score([Staff([Note("c'4")])])

>>> score
Score<<1>>

>>> scoretools.all_are_scores([score])
True
```

True when *expr* is an empty sequence:

```
>>> scoretools.all_are_scores([])
True
```

Otherwise false:

```
>>> scoretools.all_are_scores('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

`scoretools.get_first_score_in_improper_parentage_of_component`

`scoretools.get_first_score_in_improper_parentage_of_component` (*component*)

New in version 2.0. Get first score in improper parentage of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score = Score([staff])

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

>>> scoretools.get_first_score_in_improper_parentage_of_component(score.leaves[0])
Score<<1>>
```

Return score or none.

`scoretools.get_first_score_in_proper_parentage_of_component`

`scoretools.get_first_score_in_proper_parentage_of_component` (*component*)

New in version 2.0. Get first score in proper parentage of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> score = Score([staff])

>>> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

>>> scoretools.get_first_score_in_proper_parentage_of_component(score.leaves[0])
Score<<1>>
```

Return score or none.

`scoretools.list_performer_names`

`scoretools.list_performer_names` (*locale='en-us'*)

New in version 2.5. List performer names:

```
>>> for performer_name in scoretools.list_performer_names():
...     performer_name
...
'accordionist'
'baritone'
'bass'
'bassist'
'bassoonist'
'cellist'
'clarinetist'
'contralto'
'flutist'
'guitarist'
'harpist'
'harpsichordist'
'hornist'
'mezzo-soprano'
'oboist'
'percussionist'
'pianist'
'saxophonist'
'soprano'
'tenor'
'trombonist'
'trumpeter'
'tubist'
'vibraphonist'
'violinist'
'violist'
'xylophonist'
```

Available values for *locale* are 'en-us' and 'en-uk'.

scoretools.list_primary_performer_names

scoretools.list_primary_performer_names()

New in version 2.5. List performer names:

```
>>> for pair in scoretools.list_primary_performer_names():
...     pair
...
('accordionist', 'acc.')
('baritone', 'baritone')
('bass', 'bass')
('bassist', 'vb.')
('bassoonist', 'bsn.')
('cellist', 'vc.')
('clarinetist', 'cl. in B-flat')
('contralto', 'contralto')
('flutist', 'fl.')
('guitarist', 'gt.')
('harpist', 'hp.')
('harpsichordist', 'hpschd.')
('hornist', 'hn.')
('mezzo-soprano', 'mezzo-soprano')
('oboist', 'ob.')
('pianist', 'pf.')
```

```
( 'saxophonist', 'alto sax.')
( 'soprano', 'soprano')
( 'tenor', 'tenor')
( 'trombonist', 'ten. trb.')
( 'trumpeter', 'tp.')
( 'tubist', 'tb.')
( 'violinist', 'vn.')
( 'violist', 'va.')
```

Return list.

scoretools.make_empty_piano_score

scoretools.**make_empty_piano_score**()

New in version 1.1. Make empty piano score:

```
>>> score, treble, bass = scoretools.make_empty_piano_score()

>>> f(score)
\new Score <<
  \new PianoStaff <<
    \context Staff = "treble" {
      \clef "treble"
    }
    \context Staff = "bass" {
      \clef "bass"
    }
  >>
>>
```

Return score, treble staff, bass staff. Changed in version 2.0: renamed scoretools.make_piano_staff() to scoretools.make_empty_piano_score().

scoretools.make_piano_score_from_leaves

scoretools.**make_piano_score_from_leaves** (*leaves*, *lowest_treble_pitch=None*)

New in version 2.0. Make piano score from *leaves*:

```
>>> notes = [Note(x, (1, 4)) for x in [-12, 37, -10, 2, 4, 17]]
>>> score, treble_staff, bass_staff = scoretools.make_piano_score_from_leaves(notes)

>>> f(score)
\new Score <<
  \new PianoStaff <<
    \context Staff = "treble" {
      \clef "treble"
      r4
      cs''''4
      r4
      d'4
      e'4
      f''4
    }
    \context Staff = "bass" {
      \clef "bass"
      c4
    }
  >>
>>
```

```

        r4
        d4
        r4
        r4
        r4
    }
>>
>>

```

When `lowest_treble_pitch=None` set to B3.

Return score, treble staff, bass staff.

`scoretools.make_piano_sketch_score_from_leaves`

`scoretools.make_piano_sketch_score_from_leaves` (*leaves*, *lowest_treble_pitch=None*)

New in version 2.0. Make piano sketch score from *leaves*:

```

>>> notes = notetools.make_notes([-12, -10, -8, -7, -5, 0, 2, 4, 5, 7], [(1, 4)])
>>> score, treble_staff, bass_staff = scoretools.make_piano_sketch_score_from_leaves(notes)

>>> f(score)
\new Score \with {
  \override BarLine #'stencil = ##f
  \override BarNumber #'transparent = ##t
  \override SpanBar #'stencil = ##f
  \override TimeSignature #'stencil = ##f
} <<
  \new PianoStaff <<
    \context Staff = "treble" {
      \clef "treble"
      #(set-accidental-style 'forget)
      r4
      r4
      r4
      r4
      r4
      c'4
      d'4
      e'4
      f'4
      g'4
    }
    \context Staff = "bass" {
      \clef "bass"
      #(set-accidental-style 'forget)
      c4
      d4
      e4
      f4
      g4
      r4
      r4
      r4
      r4
    }
  }

```



```
>>
>>
```

When `lowest_treble_pitch=None` set to B3.

Make time signatures and bar numbers transparent.

Do not print bar lines or span bars.

Set all staff accidental styles to forget.

Return score, treble staff, bass staff.

sequencetools

concrete classes

sequencetools.CyclicList

`abjad.tools.sequencetools.CyclicList`

class sequencetools.CyclicList

New in version 2.0. Abjad model of cyclic list:

```
>>> from abjad.tools import sequencetools

>>> cyclic_list = sequencetools.CyclicList('abcd')

>>> cyclic_list
CyclicList([a, b, c, d])

>>> for x in range(8):
...     print x, cyclic_list[x]
...
0 a
1 b
2 c
3 d
4 a
5 b
6 c
7 d
```

Cyclic lists overload the item-getting method of built-in lists.

Cyclic lists return a value for any integer index.

Cyclic lists otherwise behave exactly like built-in lists.

Methods

`CyclicList.append()`

`L.append(object)` – append object to end

Inherited from `__builtin__.list`

`CyclicList.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`CyclicList.extend()`

`L.extend(iterable)` – extend list by appending elements from the iterable

Inherited from `__builtin__.list`

`CyclicList.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`CyclicList.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`CyclicList.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`CyclicList.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`CyclicList.reverse()`

`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`CyclicList.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`CyclicList.__add__()`

`x.__add__(y)` <==> `x+y`

Inherited from `__builtin__.list`

`CyclicList.__contains__()`

`x.__contains__(y)` <==> `y in x`

Inherited from `__builtin__.list`

`CyclicList.__delitem__()`

`x.__delitem__(y)` <==> `del x[y]`

Inherited from `__builtin__.list`

`CyclicList.__delslice__()`

`x.__delslice__(i, j)` <==> `del x[i:j]`

Use of negative indices is not supported.

Inherited from `__builtin__.list`

```

CyclicList.__eq__()
    x.__eq__(y) <==> x==y
    Inherited from __builtin__.list

CyclicList.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.list

CyclicList.__getitem__(expr)

CyclicList.__getslice__(start_index, stop_index)

CyclicList.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.list

CyclicList.__iadd__()
    x.__iadd__(y) <==> x+=y
    Inherited from __builtin__.list

CyclicList.__imul__()
    x.__imul__(y) <==> x*=y
    Inherited from __builtin__.list

CyclicList.__iter__() <==> iter(x)
    Inherited from __builtin__.list

CyclicList.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.list

CyclicList.__len__() <==> len(x)
    Inherited from __builtin__.list

CyclicList.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.list

CyclicList.__mul__()
    x.__mul__(n) <==> x*n
    Inherited from __builtin__.list

CyclicList.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.list

CyclicList.__repr__()

CyclicList.__reversed__()
    L.__reversed__() – return a reverse iterator over the list
    Inherited from __builtin__.list

CyclicList.__rmul__()
    x.__rmul__(n) <==> n*x
    Inherited from __builtin__.list

```

```
CyclicList.__setitem__()
    x.__setitem__(i, y) <==> x[i]=y

    Inherited from __builtin__.list

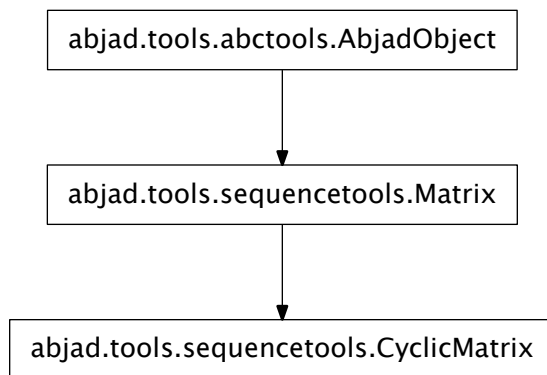
CyclicList.__setslice__()
    x.__setslice__(i, j, y) <==> x[i:j]=y

    Use of negative indices is not supported.

    Inherited from __builtin__.list

CyclicList.__str__()
```

sequencetools.CyclicMatrix



```
class sequencetools.CyclicMatrix(*args, **kwargs)
    New in version 2.0. Abjad model of cyclic matrix.

    Initialize from rows:

    >>> from abjad.tools import sequencetools

    >>> cyclic_matrix = sequencetools.CyclicMatrix(
    ...     [[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

    >>> cyclic_matrix
    CyclicMatrix(3x4)

    >>> cyclic_matrix[2]
    CyclicTuple([20, 21, 22, 23])

    >>> cyclic_matrix[2][2]
    22

    >>> cyclic_matrix[99]
    CyclicTuple([0, 1, 2, 3])
```

```
>>> cyclic_matrix[99][99]
3
```

Initialize from columns:

```
>>> cyclic_matrix = sequencetools.CyclicMatrix(
...     columns=[[0, 10, 20], [1, 11, 21], [2, 12, 22], [3, 13, 23]])
```

```
>>> cyclic_matrix
CyclicMatrix(3x4)
```

```
>>> cyclic_matrix[2]
CyclicTuple([20, 21, 22, 23])
```

```
>>> cyclic_matrix[2][2]
22
```

```
>>> cyclic_matrix[99]
CyclicTuple([0, 1, 2, 3])
```

```
>>> cyclic_matrix[99][99]
3
```

CyclicMatrix implements only item retrieval in this revision.

Concatenation and division remain to be implemented.

Standard transforms of linear algebra remain to be implemented.

Read-only Properties

CyclicMatrix.columns

Read-only columns:

```
>>> cyclic_matrix = sequencetools.CyclicMatrix(
...     [[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])
```

```
>>> cyclic_matrix.columns
CyclicTuple([[0, 10, 20], [1, 11, 21], [2, 12, 22], [3, 13, 23]])
```

Return cyclic tuple.

CyclicMatrix.rows

Read-only rows:

```
>>> cyclic_matrix = sequencetools.CyclicMatrix(
...     [[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])
```

```
>>> cyclic_matrix.rows
CyclicTuple([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])
```

Return cyclic tuple.

CyclicMatrix.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`CyclicMatrix.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`CyclicMatrix.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CyclicMatrix.__getitem__(expr)`

`CyclicMatrix.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`CyclicMatrix.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CyclicMatrix.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

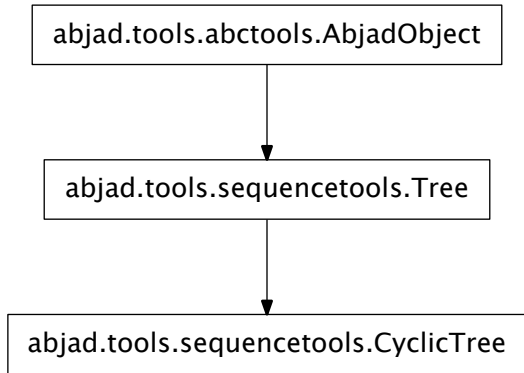
`CyclicMatrix.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`CyclicMatrix.__repr__()`

sequencetools.CyclicTree

class `sequencetools.CyclicTree` (*expr*)

New in version 2.5. Like `Tree` but with cyclic s Abjad data structure to work with a sequence whose elements have been grouped into arbitrarily many levels of **cyclic** containment.

Exactly like the `Tree` class but with the additional affordance that all integer indices of any size work at every level of structure; like `CyclicTuple`, `CyclicList` and `CyclicMatrix`, no index errors raises in working with objects of this class.

```
>>> from abjad.tools import sequencetools
```

Here is a cyclic tree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> cyclic_tree = sequencetools.CyclicTree(sequence)
```

```
>>> cyclic_tree
CyclicTree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

Here's an internal node:

```
>>> cyclic_tree[2]
CyclicTree([4, 5])
```

Here's the same node indexed with a different way:

```
>>> cyclic_tree[2]
CyclicTree([4, 5])
```

With a negative index:

```
>>> cyclic_tree[-2]
CyclicTree([4, 5])
```

And another negative index:

```
>>> cyclic_tree[-6]
CyclicTree([4, 5])
```

Here's a leaf node:

```
>>> cyclic_tree[2][0]
CyclicTree(4)
```

And here's the same node indexed a different way:

```
>>> cyclic_tree[2][20]
CyclicTree(4)
```

All other interface attributes function as in `Tree`.

Read-only Properties

`CyclicTree.children`

New in version 2.4. Children of node:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].children
(Tree(2), Tree(3))
```

Return tuple of zero or more nodes.

Inherited from `sequencetools.Tree`

`CyclicTree.depth`

New in version 2.4. Depth of subtree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].depth
2
```

Return nonnegative integer.

Inherited from `sequencetools.Tree`

`CyclicTree.improper_parentage`

New in version 2.4. Improper parentage of node:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].improper_parentage
(Tree([2, 3]), Tree([[0, 1], [2, 3], [4, 5], [6, 7]]))
```

Return tuple of one or more nodes.

Inherited from `sequencetools.Tree`

`CyclicTree.index_in_parent`

New in version 2.4. Index of node in parent:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].index_in_parent
1
```


Return nonnegative integer.

Inherited from `sequencetools.Tree`

`CyclicTree.level`

New in version 2.4. Level of node:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].level
1
```

Return nonnegative integer.

Inherited from `sequencetools.Tree`

`CyclicTree.negative_level`

New in version 2.4. Negative level of node:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].negative_level
-2
```

Return negative integer.

Inherited from `sequencetools.Tree`

`CyclicTree.position`

New in version 2.4. Position of node relative to root:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].position
(1,)
```

Return tuple of zero or more nonnegative integers.

Inherited from `sequencetools.Tree`

`CyclicTree.proper_parentage`

New in version 2.4. Proper parentage of node:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].proper_parentage
(Tree([[0, 1], [2, 3], [4, 5], [6, 7]]),)
```

Return tuple of zero or more nodes.

Inherited from `sequencetools.Tree`

`CyclicTree.root`

New in version 2.4. Root of tree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree[1].proper_parentage
(Tree([[0, 1], [2, 3], [4, 5], [6, 7]]),)
```

Return node.

Inherited from `sequencetools.Tree`

CyclicTree.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

CyclicTree.width

New in version 2.4. Number of leaves in subtree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree[1].width
2
```

Return nonnegative integer.

Inherited from `sequencetools.Tree`

Methods

CyclicTree.get_next_n_complete_nodes_at_level(*n*, *level*)

New in version 2.5. Get next *n* complete nodes at *level* from node.

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

With nonnegative *level*:

Get next 4 nodes at level 2:

```
>>> tree[0][0].get_next_n_complete_nodes_at_level(4, 2)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level 1:

```
>>> tree[0][0].get_next_n_complete_nodes_at_level(3, 1)
[Tree([1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7])]
```

With negative *level*:

Get next 4 nodes at level -1:

```
>>> tree[0][0].get_next_n_complete_nodes_at_level(4, -1)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level -2:

```
>>> tree[0][0].get_next_n_complete_nodes_at_level(3, -2)
[Tree([1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7])]
```

Trim first node if necessary.

Return list of nodes.

Inherited from `sequencetools.Tree`

`CyclicTree.get_next_n_nodes_at_level` (*n*, *level*)

New in version 2.4. Get next *n* nodes at *level* from node.

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

With nonnegative *level*:

Get next 4 nodes at level 2:

```
>>> tree[0][0].get_next_n_nodes_at_level(4, 2)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level 1:

```
>>> tree[0][0].get_next_n_nodes_at_level(3, 1)
[Tree([1]), Tree([2, 3]), Tree([4, 5])]
```

Get next node at level 0:

```
>>> tree[0][0].get_next_n_nodes_at_level(1, 0)
[Tree([[1], [2, 3], [4, 5], [6, 7]])]
```

With negative *level*:

Get next 4 nodes at level -1:

```
>>> tree[0][0].get_next_n_nodes_at_level(4, -1)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level -2:

```
>>> tree[0][0].get_next_n_nodes_at_level(3, -2)
[Tree([1]), Tree([2, 3]), Tree([4, 5])]
```

Trim first node if necessary.

Return list of nodes.

Inherited from `sequencetools.Tree`

`CyclicTree.get_node_at_position` (*position*)

New in version 2.4. Get node at *position*:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree.get_node_at_position((2, 1))
Tree(5)
```

Return node.

Inherited from `sequencetools.Tree`

`CyclicTree.get_position_of_descendant` (*descendant*)

New in version 2.4. Get position of *descendent* relative to node rather than relative to root:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[3].get_position_of_descendant(tree[3][0])
(0,)
```

Return tuple of zero or more nonnegative integers.

Inherited from `sequencetools.Tree`

`CyclicTree.is_at_level(level)`

New in version 2.4. True when node is at *level* in tree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1][1].is_at_level(-1)
True
```

False otherwise:

```
>>> tree[1][1].is_at_level(0)
False
```

Return boolean.

Predicate works for positive, negative and zero-valued *level*.

Inherited from `sequencetools.Tree`

`CyclicTree.iterate_at_level(level)`

New in version 2.4. Iterate depth at *level*:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> for x in tree.iterate_at_level(0): x
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])

>>> for x in tree.iterate_at_level(1): x
...
Tree([0, 1])
Tree([2, 3])
Tree([4, 5])
Tree([6, 7])

>>> for x in tree.iterate_at_level(2): x
...
Tree(0)
Tree(1)
Tree(2)
Tree(3)
Tree(4)
Tree(5)
Tree(6)
Tree(7)

>>> for x in tree.iterate_at_level(-1): x
...
Tree(0)
Tree(1)
Tree(2)
Tree(3)
Tree(4)
Tree(5)
```

```

Tree(6)
Tree(7)

>>> for x in tree.iterate_at_level(-2): x
...
Tree([0, 1])
Tree([2, 3])
Tree([4, 5])
Tree([6, 7])

>>> for x in tree.iterate_at_level(-3): x
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])

```

Return node generator.

Inherited from `sequencetools.Tree`

`CyclicTree.iterate_depth_first()`

New in version 2.4. Iterate tree depth-first:

```

>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> for node in tree.iterate_depth_first(): node
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
Tree([0, 1])
Tree(0)
Tree(1)
Tree([2, 3])
Tree(2)
Tree(3)
Tree([4, 5])
Tree(4)
Tree(5)
Tree([6, 7])
Tree(6)
Tree(7)

```

Return node generator.

Inherited from `sequencetools.Tree`

`CyclicTree.iterate_payload()`

New in version 2.4. Iterate tree payload:

```

>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> for element in tree.iterate_payload():
...     element
...
0
1
2
3
4
5

```

6
7

Return payload generator.

Inherited from `sequencetools.Tree`

`CyclicTree.remove(node)`

New in version 2.4. Remove *node* from tree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree.remove(tree[1])

>>> tree
Tree([[0, 1], [4, 5], [6, 7]])
```

Return none.

Inherited from `sequencetools.Tree`

`CyclicTree.remove_to_root()`

New in version 2.4. Remove node and all nodes left of node to root:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]

>>> tree = sequencetools.Tree(sequence)
>>> tree[0][0].remove_to_root()
>>> tree
Tree([[1], [2, 3], [4, 5], [6, 7]])

>>> tree = sequencetools.Tree(sequence)
>>> tree[0][1].remove_to_root()
>>> tree
Tree([[2, 3], [4, 5], [6, 7]])

>>> tree = sequencetools.Tree(sequence)
>>> tree[1].remove_to_root()
>>> tree
Tree([[4, 5], [6, 7]])
```

Modify in-place to root.

Return none.

Inherited from `sequencetools.Tree`

`CyclicTree.to_nested_lists()`

New in version 2.5. Change tree to nested lists:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])

>>> tree.to_nested_lists()
[[0, 1], [2, 3], [4, 5], [6, 7]]
```

Return list of lists.

Inherited from `sequencetools.Tree`

Special Methods

`CyclicTree.__contains__(expr)`
 Inherited from `sequencetools.Tree`

`CyclicTree.__eq__(other)`
 Inherited from `sequencetools.Tree`

`CyclicTree.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`CyclicTree.__getitem__(expr)`
 Inherited from `sequencetools.Tree`

`CyclicTree.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`CyclicTree.__iter__()`

`CyclicTree.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`CyclicTree.__len__()`
 Inherited from `sequencetools.Tree`

`CyclicTree.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`CyclicTree.__ne__(other)`
 Inherited from `sequencetools.Tree`

`CyclicTree.__repr__()`
 Inherited from `sequencetools.Tree`

`CyclicTree.__str__()`
 Inherited from `sequencetools.Tree`

`sequencetools.CyclicTuple`

`abjad.tools.sequencetools.CyclicTuple`

class sequencetools.CyclicTuple

New in version 2.0. Abjad model of cyclic tuple:

```
>>> from abjad.tools import sequencetools

>>> cyclic_tuple = sequencetools.CyclicTuple('abcd')

>>> cyclic_tuple
CyclicTuple([a, b, c, d])

>>> for x in range(8):
...     print x, cyclic_tuple[x]
...
0 a
1 b
2 c
3 d
4 a
5 b
6 c
7 d
```

Cyclic tuples overload the item-getting method of built-in tuples.

Cyclic tuples return a value for any integer index.

Cyclic tuples otherwise behave exactly like built-in tuples.

Methods

CyclicTuple.**count**(*value*) → integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

CyclicTuple.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

Special Methods

CyclicTuple.**__add__**()

`x.__add__(y) <==> x+y`

Inherited from `__builtin__.tuple`

CyclicTuple.**__contains__**()

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

CyclicTuple.**__eq__**()

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

CyclicTuple.**__ge__**()

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

CyclicTuple.**__getitem__**(*expr*)

CyclicTuple.**__getslice__**(*start_index*, *stop_index*)


```
CyclicTuple.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.tuple

CyclicTuple.__hash__() <==> hash(x)
    Inherited from __builtin__.tuple

CyclicTuple.__iter__() <==> iter(x)
    Inherited from __builtin__.tuple

CyclicTuple.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.tuple

CyclicTuple.__len__() <==> len(x)
    Inherited from __builtin__.tuple

CyclicTuple.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple

CyclicTuple.__mul__()
    x.__mul__(n) <==> x*n
    Inherited from __builtin__.tuple

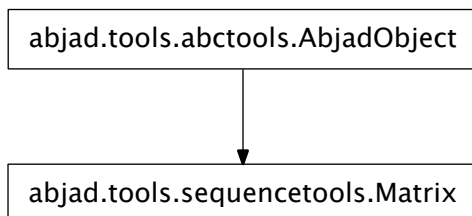
CyclicTuple.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple

CyclicTuple.__repr__()

CyclicTuple.__rmul__()
    x.__rmul__(n) <==> n*x
    Inherited from __builtin__.tuple

CyclicTuple.__str__()
```

sequencetools.Matrix



```
class sequencetools.Matrix(*args, **kwargs)
    New in version 2.0. Abjad model of matrix.
```

Initialize from rows:

```
>>> matrix = sequencetools.Matrix([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

>>> matrix
Matrix(3x4)

>>> matrix[:]
((0, 1, 2, 3), (10, 11, 12, 13), (20, 21, 22, 23))

>>> matrix[2]
(20, 21, 22, 23)

>>> matrix[2][0]
20
```

Initialize from columns:

```
>>> matrix = sequencetools.Matrix(
...     columns=[[0, 10, 20], [1, 11, 21], [2, 12, 22], [3, 13, 23]])

>>> matrix
Matrix(3x4)

>>> matrix[:]
((0, 1, 2, 3), (10, 11, 12, 13), (20, 21, 22, 23))

>>> matrix[2]
(20, 21, 22, 23)

>>> matrix[2][0]
20
```

Matrix implements only item retrieval in this revision.

Concatenation and division remain to be implemented.

Standard transforms of linear algebra remain to be implemented.

Read-only Properties

`Matrix.columns`

Read-only columns:

```
>>> matrix = sequencetools.Matrix([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

>>> matrix.columns
((0, 10, 20), (1, 11, 21), (2, 12, 22), (3, 13, 23))
```

Return tuple.

`Matrix.rows`

Read-only rows:

```
>>> matrix = sequencetools.Matrix([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

>>> matrix.rows
((0, 1, 2, 3), (10, 11, 12, 13), (20, 21, 22, 23))
```

Return tuple.

`Matrix.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`Matrix.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Matrix.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Matrix.__getitem__(expr)`

`Matrix.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Matrix.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Matrix.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

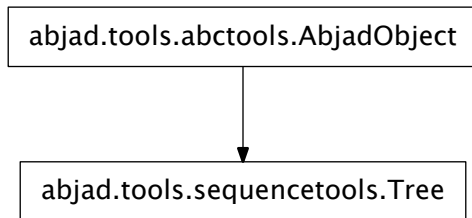
`Matrix.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Matrix.__repr__()`

sequencetools.Tree

class `sequencetools.Tree`(*expr*)

New in version 2.4. Abjad data structure to work with a sequence whose elements have been grouped into arbitrarily many levels of containment.

Example: a list of pitches that have been grouped into cells that have, in turn, been grouped into groups of cells that have, in turn, been grouped into groups of groups of cells.

```
>>> from abjad.tools import sequencetools
```

Here is a tree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])

>>> tree.parent is None
True

>>> tree.children
(Tree([0, 1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7]))

>>> tree.depth
3
```

Here's an internal node:

```
>>> tree[2]
Tree([4, 5])

>>> tree[2].parent
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])

>>> tree[2].children
(Tree(4), Tree(5))

>>> tree[2].depth
2
```

```

>>> tree[2].level
1

Here's a leaf node:

>>> tree[2][0]
Tree(4)

>>> tree[2][0].parent
Tree([4, 5])

>>> tree[2][0].children
()

>>> tree[2][0].depth
1

>>> tree[2][0].level
2

>>> tree[2][0].position
(2, 0)

>>> tree[2][0].payload
4

```

Only leaf nodes carry payload. Internal nodes carry no payload.

Negative levels are available to work with trees bottom-up instead of top-down.

Trees do not yet implement append or extend methods.

Read-only Properties

Tree.**children**

New in version 2.4. Children of node:

```

>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].children
(Tree(2), Tree(3))

```

Return tuple of zero or more nodes.

Tree.**depth**

New in version 2.4. Depth of subtree:

```

>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].depth
2

```

Return nonnegative integer.

Tree.**improper_parentage**

New in version 2.4. Improper parentage of node:

```

>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

```

```
>>> tree[1].improper_parentage
(Tree([2, 3]), Tree([[0, 1], [2, 3], [4, 5], [6, 7]]))
```

Return tuple of one or more nodes.

Tree.**index_in_parent**

New in version 2.4. Index of node in parent:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].index_in_parent
1
```

Return nonnegative integer.

Tree.**level**

New in version 2.4. Level of node:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].level
1
```

Return nonnegative integer.

Tree.**negative_level**

New in version 2.4. Negative level of node:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].negative_level
-2
```

Return negative integer.

Tree.**position**

New in version 2.4. Position of node relative to root:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].position
(1,)
```

Return tuple of zero or more nonnegative integers.

Tree.**proper_parentage**

New in version 2.4. Proper parentage of node:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[1].proper_parentage
(Tree([[0, 1], [2, 3], [4, 5], [6, 7]]),)
```

Return tuple of zero or more nodes.

Tree.**root**

New in version 2.4. Root of tree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree[1].proper_parentage
(Tree([[0, 1], [2, 3], [4, 5], [6, 7]]),)
```

Return node.

Tree.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Tree.width

New in version 2.4. Number of leaves in subtree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree[1].width
2
```

Return nonnegative integer.

Methods

Tree.get_next_n_complete_nodes_at_level (*n*, *level*)

New in version 2.5. Get next *n* complete nodes at *level* from node.

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

With nonnegative *level*:

Get next 4 nodes at level 2:

```
>>> tree[0][0].get_next_n_complete_nodes_at_level(4, 2)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level 1:

```
>>> tree[0][0].get_next_n_complete_nodes_at_level(3, 1)
[Tree([1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7])]
```

With negative *level*:

Get next 4 nodes at level -1:

```
>>> tree[0][0].get_next_n_complete_nodes_at_level(4, -1)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level -2:

```
>>> tree[0][0].get_next_n_complete_nodes_at_level(3, -2)
[Tree([1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7])]
```

Trim first node if necessary.

Return list of nodes.

`Tree.get_next_n_nodes_at_level` (*n*, *level*)

New in version 2.4. Get next *n* nodes at *level* from node.

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

With nonnegative *level*:

Get next 4 nodes at level 2:

```
>>> tree[0][0].get_next_n_nodes_at_level(4, 2)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level 1:

```
>>> tree[0][0].get_next_n_nodes_at_level(3, 1)
[Tree([1]), Tree([2, 3]), Tree([4, 5])]
```

Get next node at level 0:

```
>>> tree[0][0].get_next_n_nodes_at_level(1, 0)
[Tree([[1], [2, 3], [4, 5], [6, 7]])]
```

With negative *level*:

Get next 4 nodes at level -1:

```
>>> tree[0][0].get_next_n_nodes_at_level(4, -1)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level -2:

```
>>> tree[0][0].get_next_n_nodes_at_level(3, -2)
[Tree([1]), Tree([2, 3]), Tree([4, 5])]
```

Trim first node if necessary.

Return list of nodes.

`Tree.get_node_at_position` (*position*)

New in version 2.4. Get node at *position*:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree.get_node_at_position((2, 1))
Tree(5)
```

Return node.

`Tree.get_position_of_descendant` (*descendant*)

New in version 2.4. Get position of *descendant* relative to node rather than relative to root:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree[3].get_position_of_descendant(tree[3][0])
(0,)
```

Return tuple of zero or more nonnegative integers.

`Tree.is_at_level` (*level*)

New in version 2.4. True when node is at *level* in tree:


```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree[1][1].is_at_level(-1)
True
```

False otherwise:

```
>>> tree[1][1].is_at_level(0)
False
```

Return boolean.

Predicate works for positive, negative and zero-valued *level*.

Tree.**iterate_at_level**(*level*)

New in version 2.4. Iterate depth at *level*:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)
```

```
>>> for x in tree.iterate_at_level(0): x
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

```
>>> for x in tree.iterate_at_level(1): x
...
Tree([0, 1])
Tree([2, 3])
Tree([4, 5])
Tree([6, 7])
```

```
>>> for x in tree.iterate_at_level(2): x
...
Tree(0)
Tree(1)
Tree(2)
Tree(3)
Tree(4)
Tree(5)
Tree(6)
Tree(7)
```

```
>>> for x in tree.iterate_at_level(-1): x
...
Tree(0)
Tree(1)
Tree(2)
Tree(3)
Tree(4)
Tree(5)
Tree(6)
Tree(7)
```

```
>>> for x in tree.iterate_at_level(-2): x
...
Tree([0, 1])
Tree([2, 3])
Tree([4, 5])
Tree([6, 7])
```

```
>>> for x in tree.iterate_at_level(-3): x
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

Return node generator.

Tree.**iterate_depth_first**()

New in version 2.4. Iterate tree depth-first:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> for node in tree.iterate_depth_first(): node
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
Tree([0, 1])
Tree(0)
Tree(1)
Tree([2, 3])
Tree(2)
Tree(3)
Tree([4, 5])
Tree(4)
Tree(5)
Tree([6, 7])
Tree(6)
Tree(7)
```

Return node generator.

Tree.**iterate_payload**()

New in version 2.4. Iterate tree payload:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> for element in tree.iterate_payload():
...     element
...
0
1
2
3
4
5
6
7
```

Return payload generator.

Tree.**remove**(node)

New in version 2.4. Remove *node* from tree:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
>>> tree = sequencetools.Tree(sequence)

>>> tree.remove(tree[1])

>>> tree
Tree([[0, 1], [4, 5], [6, 7]])
```

Return none.

Tree.**remove_to_root**()

New in version 2.4. Remove node and all nodes left of node to root:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
```

```
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree[0][0].remove_to_root()
```

```
>>> tree
```

```
Tree([[1], [2, 3], [4, 5], [6, 7]])
```

```
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree[0][1].remove_to_root()
```

```
>>> tree
```

```
Tree([[2, 3], [4, 5], [6, 7]])
```

```
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree[1].remove_to_root()
```

```
>>> tree
```

```
Tree([[4, 5], [6, 7]])
```

Modify in-place to root.

Return none.

Tree.**to_nested_lists**()

New in version 2.5. Change tree to nested lists:

```
>>> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
```

```
>>> tree = sequencetools.Tree(sequence)
```

```
>>> tree
```

```
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

```
>>> tree.to_nested_lists()
```

```
[[0, 1], [2, 3], [4, 5], [6, 7]]
```

Return list of lists.

Special Methods

Tree.**__contains__**(*expr*)

Tree.**__eq__**(*other*)

Tree.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Tree.**__getitem__**(*expr*)

Tree.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Tree.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Tree.__len__()`

`Tree.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Tree.__ne__(other)`

`Tree.__repr__()`

`Tree.__str__()`

functions

`sequencetools.all_are_assignable_integers`

`sequencetools.all_are_assignable_integers(expr)`

New in version 2.0. True when *expr* is a sequence and all elements in *expr* are notehead-assignable integers:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.all_are_assignable_integers([1, 2, 3, 4, 6, 7, 8, 12, 14, 15, 16])
True
```

True when *expr* is an empty sequence:

```
>>> sequencetools.all_are_assignable_integers([])
True
```

False otherwise:

```
>>> sequencetools.all_are_assignable_integers('foo')
False
```

Return boolean.

`sequencetools.all_are_equal`

`sequencetools.all_are_equal(expr)`

New in version 2.0. True when *expr* is a sequence and all elements in *expr* are equal:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.all_are_equal([99, 99, 99, 99, 99, 99])
True
```

True when *expr* is an empty sequence:

```
>>> sequencetools.all_are_equal([])
True
```

False otherwise:

```
>>> sequencetools.all_are_equal(17)
False
```

Return boolean.

sequencetools.all_are_integer_equivalent_exprs

sequencetools.all_are_integer_equivalent_exprs (*expr*)

New in version 2.0. True when *expr* is a sequence and all elements in *expr* are integer-equivalent expressions:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.all_are_integer_equivalent_exprs([1, '2', 3.0, Fraction(4, 1)])
True
```

Otherwise false:

```
>>> sequencetools.all_are_integer_equivalent_exprs([1, '2', 3.5, 4])
False
```

Return boolean.

sequencetools.all_are_integer_equivalent_numbers

sequencetools.all_are_integer_equivalent_numbers (*expr*)

New in version 2.0. True when *expr* is a sequence and all elements in *expr* are integer-equivalent numbers:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.all_are_integer_equivalent_numbers([1, 2, 3.0, Fraction(4, 1)])
True
```

Otherwise false:

```
>>> sequencetools.all_are_integer_equivalent_numbers([1, 2, 3.5, 4])
False
```

Return boolean.

sequencetools.all_are_nonnegative_integer_equivalent_numbers

sequencetools.all_are_nonnegative_integer_equivalent_numbers (*expr*)

New in version 2.0. True *expr* is a sequence and when all elements in *expr* are nonnegative integer-equivalent numbers. Otherwise false:

```
>>> from abjad.tools import sequencetools

>>> expr = [0, 0.0, Fraction(0), 2, 2.0, Fraction(2)]
>>> sequencetools.all_are_nonnegative_integer_equivalent_numbers(expr)
True
```

Return boolean.

sequencetools.all_are_nonnegative_integer_powers_of_two

`sequencetools.all_are_nonnegative_integer_powers_of_two(expr)`

New in version 2.0. True when *expr* is a sequence and all elements in *expr* are nonnegative integer powers of two:

```
>>> sequencetools.all_are_nonnegative_integer_powers_of_two([0, 1, 1, 1, 2, 4, 32, 32])
True
```

True when *expr* is an empty sequence:

```
>>> sequencetools.all_are_nonnegative_integer_powers_of_two([])
True
```

False otherwise:

```
>>> sequencetools.all_are_nonnegative_integer_powers_of_two(17)
False
```

Return boolean.

sequencetools.all_are_nonnegative_integers

`sequencetools.all_are_nonnegative_integers(expr)`

New in version 2.0. True when *expr* is a sequence and all elements in *expr* are nonnegative integers:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.all_are_nonnegative_integers([0, 1, 2, 99])
True
```

Otherwise false:

```
>>> sequencetools.all_are_nonnegative_integers([0, 1, 2, -99])
False
```

Return boolean.

sequencetools.all_are_numbers

`sequencetools.all_are_numbers(expr)`

New in version 1.1. True when *expr* is a sequence and all elements in *expr* are numbers:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.all_are_numbers([1, 2, 3.0, Fraction(13, 8)])
True
```

True when *expr* is an empty sequence:

```
>>> sequencetools.all_are_numbers([])
True
```

False otherwise:

```
>>> sequencetools.all_are_numbers(17)
False
```

Return boolean. Changed in version 2.0: renamed `sequencetools.is_numeric()` to `sequencetools.all_are_numbers()`.

`sequencetools.all_are_pairs`

`sequencetools.all_are_pairs(expr)`

True when *expr* is a sequence whose members are all sequences of length 2:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.all_are_pairs([(1, 2), (3, 4), (5, 6), (7, 8)])
True
```

True when *expr* is an empty sequence:

```
>>> sequencetools.all_are_pairs([])
True
```

False otherwise:

```
>>> sequencetools.all_are_pairs('foo')
False
```

Return boolean.

`sequencetools.all_are_pairs_of_types`

`sequencetools.all_are_pairs_of_types(expr, first_type, second_type)`

True when *expr* is a sequence whose members are all sequences of length 2, and where the first member of each pair is an instance of *first_type* and where the second member of each pair is an instance of *second_type*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.all_are_pairs_of_types([(1., 'a'), (2.1, 'b'), (3.45, 'c')], float, str)
True
```

True when *expr* is an empty sequence:

```
>>> sequencetools.all_are_pairs_of_types([], float, str)
True
```

False otherwise:

```
>>> sequencetools.all_are_pairs_of_types('foo', float, str)
False
```

Return boolean.

`sequencetools.all_are_positive_integer_equivalent_numbers`

`sequencetools.all_are_positive_integer_equivalent_numbers(expr)`

New in version 2.0. True when *expr* is a sequence and all elements in *expr* are positive integer-equivalent numbers. Otherwise false:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.all_are_positive_integer_equivalent_numbers([Fraction(4, 2), 2.0, 2])
True
```

Return boolean.

sequencetools.all_are_positive_integers

sequencetools.all_are_positive_integers (*expr*)

New in version 2.0. True when *expr* is a sequence and all elements in *expr* are positive integers:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.all_are_positive_integers([1, 2, 3, 99])
True
```

Otherwise false:

```
>>> sequencetools.all_are_positive_integers(17)
False
```

Return boolean.

sequencetools.all_are_unequal

sequencetools.all_are_unequal (*expr*)

New in version 1.1. True when *expr* is a sequence all elements in *expr* are unequal:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.all_are_unequal([1, 2, 3, 4, 9])
True
```

True when *expr* is an empty sequence:

```
>>> sequencetools.all_are_unequal([])
True
```

False otherwise:

```
>>> sequencetools.all_are_unequal(17)
False
```

Return boolean. Changed in version 2.0: renamed `sequencetools.is_unique()` to `sequencetools.all_are_unequal()`.

sequencetools.count_length_two_runs_in_sequence

sequencetools.count_length_two_runs_in_sequence (*sequence*)

New in version 1.1. Count length-2 runs in *sequence*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.count_length_two_runs_in_sequence([0, 0, 1, 1, 1, 2, 3, 4, 5])
3
```

Return nonnegative integer. Changed in version 2.0: renamed `sequencetools.count_repetitions()` to `sequencetools.count_length_two_runs_in_sequence()`.

sequencetools.divide_sequence_elements_by_greatest_common_divisor

`sequencetools.divide_sequence_elements_by_greatest_common_divisor(sequence)`

New in version 2.0. Divide *sequence* elements by greatest common divisor:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.divide_sequence_elements_by_greatest_common_divisor([2, 2, -8, -16])
[1, 1, -4, -8]
```

Allow negative *sequence* elements.

Raise type error on noninteger *sequence* elements.

Raise not implemented error when 0 in *sequence*.

Return new *sequence* object.

sequencetools.flatten_sequence

`sequencetools.flatten_sequence(sequence, classes=None, depth=-1)`

New in version 1.1. Flatten *sequence*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.flatten_sequence([1, [2, 3, [4]], 5, [6, 7, [8]]])
[1, 2, 3, 4, 5, 6, 7, 8]
```

Flatten *sequence* to depth 1:

```
>>> sequencetools.flatten_sequence([1, [2, 3, [4]], 5, [6, 7, [8]]], depth=1)
[1, 2, 3, [4], 5, 6, 7, [8]]
```

Flatten *sequence* to depth 2:

```
>>> sequencetools.flatten_sequence([1, [2, 3, [4]], 5, [6, 7, [8]]], depth=2)
[1, 2, 3, 4, 5, 6, 7, 8]
```

Leave *sequence* unchanged.

Return newly constructed *sequence* object. Changed in version 2.0: renamed `listtools.flatten()` to `sequencetools.flatten_sequence()`.

sequencetools.flatten_sequence_at_indices

`sequencetools.flatten_sequence_at_indices(sequence, indices, classes=None, depth=-1)`

New in version 2.0. Flatten *sequence* at *indices*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.flatten_sequence_at_indices([0, 1, [2, 3, 4], [5, 6, 7]], [3])
[0, 1, [2, 3, 4], 5, 6, 7]
```

Flatten *sequence* at negative *indices*:

```
>>> sequencetools.flatten_sequence_at_indices([0, 1, [2, 3, 4], [5, 6, 7]], [-1])
[0, 1, [2, 3, 4], 5, 6, 7]
```

Leave *sequence* unchanged.

Return newly constructed *sequence* object.

`sequencetools.get_indices_of_sequence_elements_equal_to_true`

`sequencetools.get_indices_of_sequence_elements_equal_to_true(sequence)`

New in version 1.1. Get indices of *sequence* elements equal to true:

```
>>> from abjad.tools import sequencetools

::

>>> sequence = [0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1]

>>> sequencetools.get_indices_of_sequence_elements_equal_to_true(sequence)
(3, 4, 5, 9, 10, 11, 12)
```

Return newly constructed tuple of zero or more nonnegative integers. Changed in version 2.0: renamed `listtools.true_indices()` to `sequencetools.get_indices_of_sequence_elements_equal_to_true()`.

`sequencetools.get_sequence_degree_of_rotational_symmetry`

`sequencetools.get_sequence_degree_of_rotational_symmetry(sequence)`

New in version 2.0. Change *sequence* to degree of rotational symmetry:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.get_sequence_degree_of_rotational_symmetry([1, 2, 3, 4, 5, 6])
1

>>> sequencetools.get_sequence_degree_of_rotational_symmetry([1, 2, 3, 1, 2, 3])
2

>>> sequencetools.get_sequence_degree_of_rotational_symmetry([1, 2, 1, 2, 1, 2])
3

>>> sequencetools.get_sequence_degree_of_rotational_symmetry([1, 1, 1, 1, 1, 1])
6
```

Return positive integer.

`sequencetools.get_sequence_element_at_cyclic_index`

`sequencetools.get_sequence_element_at_cyclic_index(sequence, index)`

New in version 2.0. Get *sequence* element at nonnegative cyclic *index*:

```
>>> from abjad.tools import sequencetools

>>> for index in range(10):
...     print '%s\t%s' % (index, sequencetools.get_sequence_element_at_cyclic_index(
...         'string', index))
... 
```

```

0  s
1  t
2  r
3  i
4  n
5  g
6  s
7  t
8  r
9  i

```

Get *sequence* element at negative cyclic *index*:

```

>>> for index in range(1, 11):
...     print '%s\t%s' % (-index, sequencetools.get_sequence_element_at_cyclic_index(
...         'string', -index))
...
-1    g
-2    n
-3    i
-4    r
-5    t
-6    s
-7    g
-8    n
-9    i
-10   r

```

Return reference to *sequence* element.

sequencetools.get_sequence_elements_at_indices

`sequencetools.get_sequence_elements_at_indices(sequence, indices)`

New in version 2.0. Get *sequence* elements at *indices*:

```

>>> from abjad.tools import sequencetools

>>> sequencetools.get_sequence_elements_at_indices('string of text', (2, 3, 10, 12))
('r', 'i', 't', 'x')

```

Return newly constructed tuple of references to *sequence* elements.

sequencetools.get_sequence_elements_frequency_distribution

`sequencetools.get_sequence_elements_frequency_distribution(sequence)`

New in version 2.0. Get *sequence* elements frequency distribution:

```

>>> sequence = [1, 3, 3, 3, 2, 1, 1, 2, 3, 3, 1, 2]

>>> sequencetools.get_sequence_elements_frequency_distribution(sequence)
[(1, 4), (2, 3), (3, 5)]

```

Return list of element / count pairs.

`sequencetools.get_sequence_period_of_rotation`

`sequencetools.get_sequence_period_of_rotation(sequence, n)`

New in version 2.0. Change *sequence* to period of rotation:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.get_sequence_period_of_rotation([1, 2, 3, 1, 2, 3], 1)
3

>>> sequencetools.get_sequence_period_of_rotation([1, 2, 3, 1, 2, 3], 2)
3

>>> sequencetools.get_sequence_period_of_rotation([1, 2, 3, 1, 2, 3], 3)
1
```

Return positive integer.

`sequencetools.increase_sequence_elements_at_indices_by_addenda`

`sequencetools.increase_sequence_elements_at_indices_by_addenda(sequence, addenda, indices)`

New in version 1.1. Increase *sequence* by *addenda* at *indices*:

```
>>> sequence = [1, 1, 2, 3, 5, 5, 1, 2, 5, 5, 6]

>>> sequencetools.increase_sequence_elements_at_indices_by_addenda(
...     sequence, [0.5, 0.5], [0, 4, 8])
[1.5, 1.5, 2, 3, 5.5, 5.5, 1, 2, 5.5, 5.5, 6]
```

Return list. Changed in version 2.0: renamed `sequencetools.increase_at_indices()` to `sequencetools.increase_sequence_elements_at_indices_by_addenda()`.

`sequencetools.increase_sequence_elements_cyclically_by_addenda`

`sequencetools.increase_sequence_elements_cyclically_by_addenda(sequence, addenda, shield=True)`

New in version 1.1.. Increase *sequence* cyclically by *addenda*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.increase_sequence_elements_cyclically_by_addenda(
...     range(10), [10, -10], shield=False)
[10, -9, 12, -7, 14, -5, 16, -3, 18, -1]
```

Increase *sequence* cyclically by *addenda* and map nonpositive values to 1:

```
>>> sequencetools.increase_sequence_elements_cyclically_by_addenda(
...     range(10), [10, -10], shield=True)
[10, 1, 12, 1, 14, 1, 16, 1, 18, 1]
```

Return list. Changed in version 2.0: renamed `sequencetools.increase_cyclic()` to `sequencetools.increase_sequence_elements_cyclically_by_addenda()`.

sequencetools.interlace_sequences

`sequencetools.interlace_sequences(*sequences)`

New in version 1.1. Interlace *sequences*:

```
>>> from abjad.tools import sequencetools

>>> k = range(100, 103)
>>> l = range(200, 201)
>>> m = range(300, 303)
>>> n = range(400, 408)
>>> sequencetools.interlace_sequences(k, l, m, n)
[100, 200, 300, 400, 101, 301, 401, 102, 302, 402, 403, 404, 405, 406, 407]
```

Return list. Changed in version 2.0: renamed `sequencetools.interlace()` to `sequencetools.interlace_sequences()`.

sequencetools.is_fraction_equivalent_pair

`sequencetools.is_fraction_equivalent_pair(expr)`

New in version 2.9. True when *expr* is an integer-equivalent pair of numbers excluding 0 as the second term:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_fraction_equivalent_pair((2, 3))
True
```

Otherwise false:

```
>>> sequencetools.is_fraction_equivalent_pair((2, 0))
False
```

Return boolean.

sequencetools.is_integer_equivalent_n_tuple

`sequencetools.is_integer_equivalent_n_tuple(expr, n)`

New in version 2.9. True when *expr* is a tuple of *n* integer-equivalent expressions:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_integer_equivalent_n_tuple((2.0, '3', Fraction(4, 1)), 3)
True
```

Otherwise false:

```
>>> sequencetools.is_integer_equivalent_n_tuple((2.5, '3', Fraction(4, 1)), 3)
False
```

Return boolean.

sequencetools.is_integer_equivalent_pair

`sequencetools.is_integer_equivalent_pair(expr)`

New in version 2.9. True when *expr* is a pair of integer-equivalent expressions:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_integer_equivalent_pair((2.0, '3'))
True
```

Otherwise false:

```
>>> sequencetools.is_integer_equivalent_pair((2.5, '3'))
False
```

Return boolean.

sequencetools.is_integer_equivalent_singleton

sequencetools.is_integer_equivalent_singleton(*expr*)
 New in version 2.9. True when *expr* is a singleton of integer-equivalent expressions:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_integer_equivalent_singleton((2.0,))
True
```

Otherwise false:

```
>>> sequencetools.is_integer_equivalent_singleton((2.5,))
False
```

Return boolean.

sequencetools.is_integer_n_tuple

sequencetools.is_integer_n_tuple(*expr*, *n*)
 New in version 2.9. True when *expr* is an integer tuple of length *n*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_integer_n_tuple((19, 20, 21), 3)
True
```

Otherwise false:

```
>>> sequencetools.is_integer_n_tuple((19, 20, 'text'), 3)
False
```

Return boolean.

sequencetools.is_integer_pair

sequencetools.is_integer_pair(*expr*)
 New in version 2.9. True when *expr* is an integer tuple of length 2:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_integer_pair((19, 20))
True
```

Otherwise false:

```
>>> sequencetools.is_integer_pair(('some', 'text'))
False
```

Return boolean.

sequencetools.is_integer_singleton

`sequencetools.is_integer_singleton(expr)`

New in version 2.9. True when *expr* is an integer tuple of length 1:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_integer_singleton((19,))
True
```

Otherwise false:

```
>>> sequencetools.is_integer_singleton(('text',))
False
```

Return boolean.

sequencetools.is_monotonically_decreasing_sequence

`sequencetools.is_monotonically_decreasing_sequence(expr)`

New in version 2.0. True when *expr* is a sequence and the elements in *expr* decrease monotonically:

```
>>> from abjad.tools import sequencetools

>>> expr = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
>>> sequencetools.is_monotonically_decreasing_sequence(expr)
True

>>> expr = [3, 3, 3, 3, 3, 3, 3, 2, 1, 0]
>>> sequencetools.is_monotonically_decreasing_sequence(expr)
True

>>> expr = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
>>> sequencetools.is_monotonically_decreasing_sequence(expr)
True
```

False when *expr* is a sequence and the elements in *expr* do not decrease monotonically:

```
>>> expr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> sequencetools.is_monotonically_decreasing_sequence(expr)
False

>>> expr = [0, 1, 2, 3, 3, 3, 3, 3, 3, 3]
>>> sequencetools.is_monotonically_decreasing_sequence(expr)
False
```

True when *expr* is a sequence and *expr* is empty:

```
>>> expr = []
>>> sequencetools.is_monotonically_decreasing_sequence(expr)
True
```

False when *expr* is not a sequence:

```
>>> sequencetools.is_monotonically_decreasing_sequence(17)
False
```

Return boolean.

sequencetools.is_monotonically_increasing_sequence

`sequencetools.is_monotonically_increasing_sequence(expr)`

New in version 2.0. True when *expr* is a sequence and the elements in *expr* increase monotonically:

```
>>> from abjad.tools import sequencetools

>>> expr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> sequencetools.is_monotonically_increasing_sequence(expr)
True

>>> expr = [0, 1, 2, 3, 3, 3, 3, 3, 3, 3]
>>> sequencetools.is_monotonically_increasing_sequence(expr)
True

>>> expr = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
>>> sequencetools.is_monotonically_increasing_sequence(expr)
True
```

False when *expr* is a sequence and the elements in *expr* do not increase monotonically:

```
>>> expr = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
>>> sequencetools.is_monotonically_increasing_sequence(expr)
False

>>> expr = [3, 3, 3, 3, 3, 3, 3, 2, 1, 0]
>>> sequencetools.is_monotonically_increasing_sequence(expr)
False
```

True when *expr* is a sequence and *expr* is empty:

```
>>> expr = []
>>> sequencetools.is_monotonically_increasing_sequence(expr)
True
```

False when *expr* is not a sequence:

```
>>> sequencetools.is_monotonically_increasing_sequence(17)
False
```

Return boolean.

sequencetools.is_n_tuple

`sequencetools.is_n_tuple(expr, n)`

True when *expr* is a tuple of length *n*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_n_tuple((19, 20, 21), 3)
True
```


Otherwise false:

```
>>> sequencetools.is_n_tuple((19, 20, 21), 4)
False
```

Return boolean.

sequencetools.is_null_tuple

`sequencetools.is_null_tuple(expr)`

New in version 2.9. True when *expr* is a tuple of length 0:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.is_null_tuple(())
True
```

Otherwise false:

```
>>> sequencetools.is_null_tuple((19, 20, 21))
False
```

Return boolean.

sequencetools.is_pair

`sequencetools.is_pair(expr)`

New in version 2.9. True when *expr* is a tuple of length 2:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.is_pair((19, 20))
True
```

Otherwise false:

```
>>> sequencetools.is_pair((19, 20, 21))
False
```

Return boolean.

sequencetools.is_permutation

`sequencetools.is_permutation(expr, length=None)`

New in version 2.0. True when *expr* is a permutation:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.is_permutation([4, 5, 0, 3, 2, 1])
True
```

Otherwise false:

```
>>> sequencetools.is_permutation([1, 1, 5, 3, 2, 1])
False
```

True when *expr* is a permutation of first *length* nonnegative integers:

```
>>> sequencetools.is_permutation([4, 5, 0, 3, 2, 1], length=6)
True
```

Otherwise false:

```
>>> sequencetools.is_permutation([4, 0, 3, 2, 1], length=6)
False
```

Return boolean.

sequencetools.is_repetition_free_sequence

sequencetools.is_repetition_free_sequence (*expr*)

New in version 2.0. True when *expr* is a sequence and *expr* is repetition free:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_repetition_free_sequence([0, 1, 2, 6, 7, 8])
True
```

False when *expr* is a sequence and *expr* is not repetition free:

```
>>> sequencetools.is_repetition_free_sequence([0, 1, 2, 2, 7, 8])
False
```

True when *expr* is an empty sequence:

```
>>> sequencetools.is_repetition_free_sequence([])
True
```

False *expr* is not a sequence:

```
>>> sequencetools.is_repetition_free_sequence(17)
False
```

Return boolean.

sequencetools.is_restricted_growth_function

sequencetools.is_restricted_growth_function (*expr*)

New in version 2.0. True when *expr* is a sequence and *expr* meets the criteria for a restricted growth function:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.is_restricted_growth_function([1, 1, 1, 1])
True

>>> sequencetools.is_restricted_growth_function([1, 1, 1, 2])
True

>>> sequencetools.is_restricted_growth_function([1, 1, 2, 1])
True

>>> sequencetools.is_restricted_growth_function([1, 1, 2, 2])
True
```

Otherwise false:

```
>>> sequencetools.is_restricted_growth_function([1, 1, 1, 3])
False
```

```
>>> sequencetools.is_restricted_growth_function(17)
False
```

A restricted growth function is a sequence l such that $l[0] == 1$ and such that $l[i] \leq \max(l[:i]) + 1$ for $1 \leq i \leq \text{len}(l)$.

Return boolean.

sequencetools.is_singleton

`sequencetools.is_singleton(expr)`

New in version 2.9. True when *expr* is a tuple of length 1:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.is_singleton((19,))
True
```

Otherwise false:

```
abjad> sequencetools.is_singleton((19, 20, 21))
False
```

Return boolean.

sequencetools.is_strictly_decreasing_sequence

`sequencetools.is_strictly_decreasing_sequence(expr)`

New in version 2.0. True when *expr* is a sequence and the elements in *expr* decrease strictly:

```
>>> from abjad.tools import sequencetools
```

```
>>> expr = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
>>> sequencetools.is_strictly_decreasing_sequence(expr)
True
```

False when *expr* is a sequence and the elements in *expr* do not decrease strictly:

```
>>> expr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> sequencetools.is_strictly_decreasing_sequence(expr)
False
```

```
>>> expr = [0, 1, 2, 3, 3, 3, 3, 3, 3, 3]
>>> sequencetools.is_strictly_decreasing_sequence(expr)
False
```

```
>>> expr = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
>>> sequencetools.is_strictly_decreasing_sequence(expr)
False
```

```
>>> expr = [3, 3, 3, 3, 3, 3, 3, 2, 1, 0]
>>> sequencetools.is_strictly_decreasing_sequence(expr)
False
```

True when *expr* is an empty sequence:

```
>>> sequencetools.is_strictly_decreasing_sequence([])
True
```

False *expr* is not a sequence:

```
>>> sequencetools.is_strictly_decreasing_sequence(17)
False
```

Return boolean.

sequencetools.is_strictly_increasing_sequence

sequencetools.is_strictly_increasing_sequence(*expr*)

New in version 2.0. True when *expr* is a sequence and the elements in *expr* increase strictly:

```
>>> from abjad.tools import sequencetools

>>> expr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> sequencetools.is_strictly_increasing_sequence(expr)
True
```

False when *expr* is a sequence and the elements in *expr* do not increase strictly:

```
>>> expr = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
>>> sequencetools.is_strictly_increasing_sequence(expr)
False

>>> expr = [3, 3, 3, 3, 3, 3, 3, 2, 1, 0]
>>> sequencetools.is_strictly_increasing_sequence(expr)
False

>>> expr = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
>>> sequencetools.is_strictly_increasing_sequence(expr)
False

>>> expr = [0, 1, 2, 3, 3, 3, 3, 3, 3, 3]
>>> sequencetools.is_strictly_increasing_sequence(expr)
False
```

True when *expr* is an empty sequence:

```
>>> sequencetools.is_strictly_increasing_sequence([])
True
```

False when *expr* is not a sequence:

```
>>> sequencetools.is_strictly_increasing_sequence(17)
False
```

Return boolean.

sequencetools.iterate_sequence_cyclically

sequencetools.iterate_sequence_cyclically(*sequence*, *step*=1, *start*=0, *length*='inf')

New in version 1.1. Iterate *sequence* cyclically according to *step*, *start* and *length*:

```
>>> from abjad.tools import sequencetools

>>> sequence = [1, 2, 3, 4, 5, 6, 7]

>>> list(sequencetools.iterate_sequence_cyclically(sequence, length=20))
[1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6]

>>> list(sequencetools.iterate_sequence_cyclically(sequence, 2, length=20))
[1, 3, 5, 7, 2, 4, 6, 1, 3, 5, 7, 2, 4, 6, 1, 3, 5, 7, 2, 4]

>>> list(sequencetools.iterate_sequence_cyclically(sequence, 2, 3, length=20))
[4, 6, 1, 3, 5, 7, 2, 4, 6, 1, 3, 5, 7, 2, 4, 6, 1, 3, 5, 7]

>>> list(sequencetools.iterate_sequence_cyclically(sequence, -2, 5, length=20))
[6, 4, 2, 7, 5, 3, 1, 6, 4, 2, 7, 5, 3, 1, 6, 4, 2, 7, 5, 3]
```

Changed in version 2.0: allows generator input.

```
>>> list(sequencetools.iterate_sequence_cyclically(xrange(1, 8), -2, 5, length=20))
[6, 4, 2, 7, 5, 3, 1, 6, 4, 2, 7, 5, 3, 1, 6, 4, 2, 7, 5, 3]
```

Set *step* to jump size and direction across sequence.

Set *start* to the index of *sequence* where the function begins iterating.

Set *length* to number of elements to return. Set to 'inf' to return infinitely.

Return generator. Changed in version 2.0: renamed `sequencetools.phasor()` to `sequencetools.iterate_sequence_cyclically()`.

`sequencetools.iterate_sequence_cyclically_from_start_to_stop`

```
sequencetools.iterate_sequence_cyclically_from_start_to_stop(sequence, start, stop)
```

New in version 1.1. Iterate *sequence* cyclically from *start* to *stop*:

```
>>> from abjad.tools import sequencetools

>>> list(sequencetools.iterate_sequence_cyclically_from_start_to_stop(range(20), 18, 10))
[18, 19, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Return generator of references to *sequence* elements. Changed in version 2.0: renamed `sequencetools.get_cyclic()` to `sequencetools.iterate_sequence_cyclically_from_start_to_stop()`.

`sequencetools.iterate_sequence_forward_and_backward_nonoverlapping`

```
sequencetools.iterate_sequence_forward_and_backward_nonoverlapping(sequence)
```

New in version 2.0. Iterate *sequence* first forward and then backward, with first and last elements repeated:

```
>>> list(sequencetools.iterate_sequence_forward_and_backward_nonoverlapping(
...     [1, 2, 3, 4, 5]))
[1, 2, 3, 4, 5, 5, 4, 3, 2, 1]
```

Return generator.

sequencetools.iterate_sequence_forward_and_backward_overlapping

`sequencetools.iterate_sequence_forward_and_backward_overlapping(sequence)`

New in version 2.0. Iterate *sequence* first forward and then backward, with first and last elements appearing only once:

```
>>> from abjad.tools import sequencetools

>>> list(sequencetools.iterate_sequence_forward_and_backward_overlapping([1, 2, 3, 4, 5]))
[1, 2, 3, 4, 5, 4, 3, 2]
```

Return generator.

sequencetools.iterate_sequence_nwise_cyclic

`sequencetools.iterate_sequence_nwise_cyclic(sequence, n)`

New in version 2.0. Iterate elements in *sequence* cyclically *n* at a time:

```
>>> from abjad.tools import sequencetools

>>> g = sequencetools.iterate_sequence_nwise_cyclic(range(6), 3)
>>> for n in range(10):
...     print g.next()
(0, 1, 2)
(1, 2, 3)
(2, 3, 4)
(3, 4, 5)
(4, 5, 0)
(5, 0, 1)
(0, 1, 2)
(1, 2, 3)
(2, 3, 4)
(3, 4, 5)
```

Return generator.

sequencetools.iterate_sequence_nwise_strict

`sequencetools.iterate_sequence_nwise_strict(sequence, n)`

New in version 2.0. Iterate elements in *sequence* *n* at a time:

```
>>> for x in sequencetools.iterate_sequence_nwise_strict(range(10), 4):
...     x
...
(0, 1, 2, 3)
(1, 2, 3, 4)
(2, 3, 4, 5)
(3, 4, 5, 6)
(4, 5, 6, 7)
(5, 6, 7, 8)
(6, 7, 8, 9)
```

Return generator.

sequencetools.iterate_sequence_nwise_wrapped

`sequencetools.iterate_sequence_nwise_wrapped(sequence, n)`

New in version 2.0. Iterate elements in *sequence* *n* at a time wrapped to beginning:

```
>>> from abjad.tools import sequencetools

>>> list(sequencetools.iterate_sequence_nwise_wrapped(range(6), 3))
[(0, 1, 2), (1, 2, 3), (2, 3, 4), (3, 4, 5), (4, 5, 0), (5, 0, 1)]
```

Return generator.

sequencetools.iterate_sequence_pairwise_cyclic

`sequencetools.iterate_sequence_pairwise_cyclic(sequence)`

New in version 1.1. Iterate *sequence* pairwise cyclic:

```
>>> from abjad.tools import sequencetools

>>> generator = sequencetools.iterate_sequence_pairwise_cyclic(range(6))

>>> generator.next()
(0, 1)
>>> generator.next()
(1, 2)
>>> generator.next()
(2, 3)
>>> generator.next()
(3, 4)
>>> generator.next()
(4, 5)
>>> generator.next()
(5, 0)
>>> generator.next()
(0, 1)
>>> generator.next()
(1, 2)
```

Return pair generator.

sequencetools.iterate_sequence_pairwise_strict

`sequencetools.iterate_sequence_pairwise_strict(sequence)`

New in version 1.1. Iterate *sequence* pairwise strict:

```
>>> from abjad.tools import sequencetools

>>> list(sequencetools.iterate_sequence_pairwise_strict(range(6)))
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)]
```

Return pair generator.

sequencetools.iterate_sequence_pairwise_wrapped`sequencetools.iterate_sequence_pairwise_wrapped(sequence)`

New in version 1.1. Iterate *sequence* pairwise wrapped:

```
>>> from abjad.tools import sequencetools

>>> list(sequencetools.iterate_sequence_pairwise_wrapped(range(6)))
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 0)]
```

Return pair generator.

sequencetools.join_subsequences`sequencetools.join_subsequences(sequence)`

New in version 2.4. Join subsequences in *sequence*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.join_subsequences([(1, 2, 3), (), (4, 5), (), (6,)])
(1, 2, 3, 4, 5, 6)
```

Return newly constructed object of subsequence type.

sequencetools.join_subsequences_by_sign_of_subsequence_elements`sequencetools.join_subsequences_by_sign_of_subsequence_elements(sequence)`

New in version 1.1. Join subsequences in *sequence* by sign:

```
>>> from abjad.tools import sequencetools

>>> sequence = [[1, 2], [3, 4], [-5, -6, -7], [-8, -9, -10], [11, 12]]
>>> sequencetools.join_subsequences_by_sign_of_subsequence_elements(sequence)
[[1, 2, 3, 4], [-5, -6, -7, -8, -9, -10], [11, 12]]

>>> sequence = [[1, 2], [], [], [3, 4, 5], [6, 7]]
>>> sequencetools.join_subsequences_by_sign_of_subsequence_elements(sequence)
[[1, 2], [], [3, 4, 5, 6, 7]]
```

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.join_sublists_by_sign()` to `sequencetools.join_subsequences_by_sign_of_subsequence_elements()`

sequencetools.map_sequence_elements_to_canonic_tuples`sequencetools.map_sequence_elements_to_canonic_tuples(sequence, big_endian=True)`

New in version 1.1. Partition *sequence* elements into canonic big-endian parts:

```
>>> sequencetools.map_sequence_elements_to_canonic_tuples(
...     range(10))
[(0,), (1,), (2,), (3,), (4,), (4, 1), (6,), (7,), (8,), (8, 1)]
```

Partition *sequence* elements into canonic little-endian parts:


```
>>> sequencetools.map_sequence_elements_to_canonic_tuples(
...     range(10), big_endian=False)
[(0,), (1,), (2,), (3,), (4,), (1, 4), (6,), (7,), (8,), (1, 8)]
```

Raise type error when *sequence* is not a list.

Raise value error on noninteger elements in *sequence*.

Return list of tuples. Changed in version 2.0: renamed `sequencetools.partition_elements_into_canonic_parts` to `sequencetools.map_sequence_elements_to_canonic_tuples()`.

sequencetools.map_sequence_elements_to_numbered_sublists

`sequencetools.map_sequence_elements_to_numbered_sublists(sequence)`

New in version 1.1. Map *sequence* elements to numbered sublists:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.map_sequence_elements_to_numbered_sublists([1, 2, -3, -4, 5])
[[1], [2, 3], [-4, -5, -6], [-7, -8, -9, -10], [11, 12, 13, 14, 15]]

>>> sequencetools.map_sequence_elements_to_numbered_sublists([1, 0, -3, -4, 5])
[[1], [], [-2, -3, -4], [-5, -6, -7, -8], [9, 10, 11, 12, 13]]
```

Note that numbering starts at 1.

Return newly constructed list of lists. Changed in version 2.0: renamed `sequencetools.lengths_to_counts()` to `sequencetools.map_sequence_elements_to_numbered_sublists()`.

sequencetools.merge_duration_sequences

`sequencetools.merge_duration_sequences(*sequences)`

Merge duration *sequences*:

```
>>> sequencetools.merge_duration_sequences([10, 10, 10], [7])
[7, 3, 10, 10]
```

Merge more duration sequences:

```
>>> sequencetools.merge_duration_sequences([10, 10, 10], [10, 10])
[10, 10, 10]
```

The idea is that each sequence element represents a duration.

Return list.

sequencetools.negate_absolute_value_of_sequence_elements_at_indices

`sequencetools.negate_absolute_value_of_sequence_elements_at_indices(sequence, indices)`

New in version 1.1. Negate the absolute value of *sequence* elements at *indices*:

```
>>> from abjad.tools import sequencetools

>>> sequence = [1, 2, 3, 4, 5, -6, -7, -8, -9, -10]
```

```
>>> sequencetools.negate_sequence_elements_at_indices(sequence, [0, 1, 2])
[-1, -2, -3, 4, 5, -6, -7, -8, -9, -10]
```

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.negate_elements_at_indices_absolutely()` to `sequencetools.negate_absolute_value_of_sequence_elements_at_indices()`.

`sequencetools.negate_absolute_value_of_sequence_elements_cyclically`

`sequencetools.negate_absolute_value_of_sequence_elements_cyclically(sequence, indices, period)`

New in version 2.0. Negate the absolute value of *sequence* elements at *indices* cyclically according to *period*:

```
>>> sequence = [1, 2, 3, 4, 5, -6, -7, -8, -9, -10]

>>> sequencetools.negate_absolute_value_of_sequence_elements_cyclically(
...     sequence, [0, 1, 2], 5)
[-1, -2, -3, 4, 5, -6, -7, -8, -9, -10]
```

Return newly constructed list.

`sequencetools.negate_sequence_elements_at_indices`

`sequencetools.negate_sequence_elements_at_indices(sequence, indices)`

New in version 1.1. Negate *sequence* elements at *indices*:

```
>>> from abjad.tools import sequencetools

>>> sequence = [1, 2, 3, 4, 5, -6, -7, -8, -9, -10]

>>> sequencetools.negate_sequence_elements_at_indices(sequence, [0, 1, 2])
[-1, -2, -3, 4, 5, -6, -7, -8, -9, -10]
```

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.negate_elements_at_indices()` to `sequencetools.negate_sequence_elements_at_indices()`.

`sequencetools.negate_sequence_elements_cyclically`

`sequencetools.negate_sequence_elements_cyclically(sequence, indices, period)`

New in version 2.0. Negate *sequence* elements at *indices* cyclically according to *period*:

```
>>> from abjad.tools import sequencetools

>>> sequence = [1, 2, 3, 4, 5, -6, -7, -8, -9, -10]

>>> sequencetools.negate_sequence_elements_cyclically(sequence, [0, 1, 2], 5)
[-1, -2, -3, 4, 5, 6, 7, 8, -9, -10]
```

Return newly constructed list.

sequencetools.overwrite_sequence_elements_at_indices

`sequencetools.overwrite_sequence_elements_at_indices` (*sequence*, *pairs*)

New in version 1.1. Overwrite *sequence* elements at indices according to *pairs*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.overwrite_sequence_elements_at_indices(range(10), [(0, 3), (5, 3)])
[0, 0, 0, 3, 4, 5, 5, 5, 8, 9]
```

Set *pairs* to a list of (anchor_index, length) pairs.

Return new list. Changed in version 2.0: renamed `sequencetools.overwrite_slices_at()` to `sequencetools.overwrite_sequence_elements_at_indices()`.

sequencetools.pair_duration_sequence_elements_with_input_pair_values

`sequencetools.pair_duration_sequence_elements_with_input_pair_values` (*duration_sequence*,
in-
put_pairs)

New in version 2.10. Pair *duration_sequence* elements with the values of *input_pairs*:

```
>>> duration_sequence = [10, 10, 10, 10]
>>> input_pairs = [('red', 1), ('orange', 18), ('yellow', 200)]

>>> sequencetools.pair_duration_sequence_elements_with_input_pair_values(
...     duration_sequence, input_pairs)
[(10, 'red'), (10, 'orange'), (10, 'yellow'), (10, 'yellow')]
```

Return a list of (element, value) output pairs.

The *input_pairs* argument must be a list of (value, duration) pairs.

The basic idea behind the function is model which input pair value is in effect at the start of each element in *duration_sequence*.

sequencetools.partition_sequence_by_backgrounded_weights

`sequencetools.partition_sequence_by_backgrounded_weights` (*sequence*, *weights*)

New in version 2.9. Partition *sequence* by backgrounded *weights*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.partition_sequence_by_backgrounded_weights(
...     [-5, -15, -10], [20, 10])
[[-5, -15], [-10]]
```

Further examples:

```
>>> sequencetools.partition_sequence_by_backgrounded_weights(
...     [-5, -15, -10], [5, 5, 5, 5, 5, 5])
[[-5], [-15], [], [], [-10], []]

>>> sequencetools.partition_sequence_by_backgrounded_weights(
...     [-5, -15, -10], [1, 29])
[[-5], [-15, -10]]
```

```
>>> sequencetools.partition_sequence_by_backgrounded_weights(
...     [-5, -15, -10], [2, 28])
[[-5], [-15, -10]]

>>> sequencetools.partition_sequence_by_backgrounded_weights(
...     [-5, -15, -10], [1, 1, 1, 1, 1, 25])
[[-5], [], [], [], [], [-15, -10]]
```

The term *backgrounded* is a short-hand concocted specifically for this function; rely on the formal definition to understand the function actually does.

Input constraint: the weight of *sequence* must equal the weight of *weights* exactly.

The signs of the elements in *sequence* are ignored.

Formal definition: partition *sequence* into *parts* such that (1.) the length of *parts* equals the length of *weights*; (2.) the elements in *sequence* appear in order in *parts*; and (3.) some final condition that is difficult to formalize.

Notionally what's going on here is that the elements of *weights* are acting as a list of successive time intervals into which the elements of *sequence* are being fit in accordance with the start offset of each *sequence* element.

The function models the grouping together of successive timespans according to which of an underlying sequence of time intervals it is in which each time span begins.

Note that, for any input to this function, the flattened output of this function always equals *sequence* exactly.

Note too that while *partition* is being used here in the sense of the other partitioning functions in the API, the distinguishing feature is this function is its ability to produce empty lists as output.

Return list of *sequence* objects.

sequencetools.partition_sequence_by_counts

`sequencetools.partition_sequence_by_counts(sequence, counts, cyclic=False, overhang=False, copy_elements=False)`

New in version 1.1. Example 1a. Partition sequence once by counts without overhang:

```
>>> sequencetools.partition_sequence_by_counts(
...     range(10), [3], cyclic=False, overhang=False)
[[0, 1, 2]]
```

Example 1b. Partition sequence once by counts without overhang:

```
>>> sequencetools.partition_sequence_by_counts(
...     range(16), [4, 3], cyclic=False, overhang=False)
[[0, 1, 2, 3], [4, 5, 6]]
```

Example 2a. Partition sequence cyclically by counts without overhang:

```
>>> sequencetools.partition_sequence_by_counts(
...     range(10), [3], cyclic=True, overhang=False)
[[0, 1, 2], [3, 4, 5], [6, 7, 8]]
```

Example 2b. Partition sequence cyclically by counts without overhang:

```
>>> sequencetools.partition_sequence_by_counts(
...     range(16), [4, 3], cyclic=True, overhang=False)
[[0, 1, 2, 3], [4, 5, 6], [7, 8, 9, 10], [11, 12, 13]]
```

Example 3a. Partition sequence once by counts with overhang:

```
>>> sequencetools.partition_sequence_by_counts(
...     range(10), [3], cyclic=False, overhang=True)
[[0, 1, 2], [3, 4, 5, 6, 7, 8, 9]]
```

Example 3b. Partition sequence once by counts with overhang:

```
>>> sequencetools.partition_sequence_by_counts(
...     range(16), [4, 3], cyclic=False, overhang=True)
[[0, 1, 2, 3], [4, 5, 6], [7, 8, 9, 10, 11, 12, 13, 14, 15]]
```

Example 4a. Partition sequence cyclically by counts with overhang:

```
>>> sequencetools.partition_sequence_by_counts(
...     range(10), [3], cyclic=True, overhang=True)
[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9]]
```

Example 4b. Partition sequence cyclically by counts with overhang:

```
>>> sequencetools.partition_sequence_by_counts(
...     range(16), [4, 3], cyclic=True, overhang=True)
[[0, 1, 2, 3], [4, 5, 6], [7, 8, 9, 10], [11, 12, 13], [14, 15]]
```

Return list of sequence objects.

sequencetools.partition_sequence_by_ratio_of_lengths

`sequencetools.partition_sequence_by_ratio_of_lengths(sequence, lengths)`

New in version 2.0. Partition *sequence* by ratio of *lengths*:

```
>>> sequence = tuple(range(10))

>>> sequencetools.partition_sequence_by_ratio_of_lengths(sequence, [1, 1, 2])
[(0, 1, 2), (3, 4), (5, 6, 7, 8, 9)]
```

Use rounding magic to avoid fractional part lengths.

Return list of *sequence* objects.

sequencetools.partition_sequence_by_ratio_of_weights

`sequencetools.partition_sequence_by_ratio_of_weights(sequence, weights)`

New in version 2.0. Partition *sequence* by ratio of *weights*:

```
>>> sequencetools.partition_sequence_by_ratio_of_weights([1] * 10, [1, 1, 1])
[[1, 1, 1], [1, 1, 1, 1], [1, 1, 1]]

>>> sequencetools.partition_sequence_by_ratio_of_weights([1] * 10, [1, 1, 1, 1])
[[1, 1, 1], [1, 1], [1, 1, 1], [1, 1]]

>>> sequencetools.partition_sequence_by_ratio_of_weights([1] * 10, [2, 2, 3])
[[1, 1, 1], [1, 1, 1], [1, 1, 1, 1]]

>>> sequencetools.partition_sequence_by_ratio_of_weights([1] * 10, [3, 2, 2])
[[1, 1, 1, 1], [1, 1, 1], [1, 1, 1]]

>>> sequencetools.partition_sequence_by_ratio_of_weights(
...     [1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2], [1, 1])
[[1, 1, 1, 1, 1, 1, 2, 2], [2, 2, 2, 2, 2]]
```

```
>>> sequencetools.partition_sequence_by_ratio_of_weights(
...     [1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2], [1, 1, 1])
[[1, 1, 1, 1, 1, 1], [2, 2, 2], [2, 2, 2]]
```

Weights of parts of returned list equal *weights_ratio* proportions with some rounding magic.

Return list of lists.

sequencetools.partition_sequence_by_restricted_growth_function

`sequencetools.partition_sequence_by_restricted_growth_function(sequence, restricted_growth_function)`

New in version 2.0. Partition *sequence* by *restricted_growth_function*:

```
>>> l = range(10)
>>> rgf = [1, 1, 2, 2, 1, 2, 3, 3, 2, 4]

>>> sequencetools.partition_sequence_by_restricted_growth_function(l, rgf)
[[0, 1, 4], [2, 3, 5, 8], [6, 7], [9]]
```

Raise value error when *sequence* length does not equal *restricted_growth_function* length.

Return list of lists.

sequencetools.partition_sequence_by_sign_of_elements

`sequencetools.partition_sequence_by_sign_of_elements(sequence, sign=[-1, 0, 1])`

New in version 1.1. Partition *sequence* elements by sign:

```
>>> sequence = [0, 0, -1, -1, 2, 3, -5, 1, 2, 5, -5, -6]

>>> list(sequencetools.partition_sequence_by_sign_of_elements(sequence))
[[0, 0], [-1, -1], [2, 3], [-5], [1, 2, 5], [-5, -6]]

>>> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign=[-1]))
[0, 0, [-1, -1], 2, 3, [-5], 1, 2, 5, [-5, -6]]

>>> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign=[0]))
[[0, 0], -1, -1, 2, 3, -5, 1, 2, 5, -5, -6]

>>> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign=[1]))
[0, 0, -1, -1, [2, 3], -5, [1, 2, 5], -5, -6]

>>> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign=[-1, 0]))
[[0, 0], [-1, -1], 2, 3, [-5], 1, 2, 5, [-5, -6]]

>>> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign=[-1, 1]))
[0, 0, [-1, -1], [2, 3], [-5], [1, 2, 5], [-5, -6]]

>>> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign=[0, 1]))
[[0, 0], -1, -1, [2, 3], -5, [1, 2, 5], -5, -6]

>>> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign=[-1, 0, 1]))
[[0, 0], [-1, -1], [2, 3], [-5], [1, 2, 5], [-5, -6]]
```

When -1 in *sign*, group negative elements.

When 0 in *sign*, group 0 elements.

When 1 in *sign*, group positive elements.

Return list of tuples of *sequence* element references. Changed in version 2.0: renamed `listtools.group_by_sign()` to `sequencetools.partition_sequence_by_sign_of_elements()`.

sequencetools.partition_sequence_by_value_of_elements

`sequencetools.partition_sequence_by_value_of_elements(sequence)`

New in version 1.1. Group *sequence* elements by value of elements:

```
>>> sequence = [0, 0, -1, -1, 2, 3, -5, 1, 1, 5, -5]

>>> sequencetools.partition_sequence_by_value_of_elements(sequence)
[(0, 0), (-1, -1), (2,), (3,), (-5,), (1, 1), (5,), (-5,)]
```

Return list of tuples of *sequence* element references. Changed in version 2.0: renamed `sequencetools.group_by_equality()` to `sequencetools.partition_sequence_by_value_of_elements()`.

sequencetools.partition_sequence_by_weights_at_least

`sequencetools.partition_sequence_by_weights_at_least(sequence, weights, cyclic=False, overhang=False)`

New in version 1.1. Partition *sequence* by *weights* at least.

```
>>> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
```

Example 1. Partition sequence once by weights at least without overhang:

```
>>> sequencetools.partition_sequence_by_weights_at_least(
...     sequence, [10, 4], cyclic=False, overhang=False)
[[3, 3, 3, 3], [4]]
```

Example 2. Partition sequence once by weights at least with overhang:

```
>>> sequencetools.partition_sequence_by_weights_at_least(
...     sequence, [10, 4], cyclic=False, overhang=True)
[[3, 3, 3, 3], [4], [4, 4, 4, 5, 5]]
```

Example 3. Partition sequence cyclically by weights at least without overhang:

```
>>> sequencetools.partition_sequence_by_weights_at_least(
...     sequence, [10, 4], cyclic=True, overhang=False)
[[3, 3, 3, 3], [4], [4, 4, 4], [5]]
```

Example 4. Partition sequence cyclically by weights at least with overhang:

```
>>> sequencetools.partition_sequence_by_weights_at_least(
...     sequence, [10, 4], cyclic=True, overhang=True)
[[3, 3, 3, 3], [4], [4, 4, 4], [5], [5]]
```

Return list of sequence objects.

`sequencetools.partition_sequence_by_weights_at_most`

`sequencetools.partition_sequence_by_weights_at_most` (*sequence, weights, cyclic=False, overhang=False*)

New in version 1.1. Partition *sequence* by *weights* at most.

```
>>> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
```

Example 1. Partition sequence once by weights at most without overhang:

```
>>> sequencetools.partition_sequence_by_weights_at_most(
...     sequence, [10, 4], cyclic=False, overhang=False)
[[3, 3, 3], [3]]
```

Example 2. Partition sequence once by weights at most with overhang:

```
>>> sequencetools.partition_sequence_by_weights_at_most(
...     sequence, [10, 4], cyclic=False, overhang=True)
[[3, 3, 3], [3], [4, 4, 4, 4, 5, 5]]
```

Example 3. Partition sequence cyclically by weights at most without overhang:

```
>>> sequencetools.partition_sequence_by_weights_at_most(
...     sequence, [10, 5], cyclic=True, overhang=False)
[[3, 3, 3], [3], [4, 4], [4], [4, 5], [5]]
```

Example 4. Partition sequence cyclically by weights at most with overhang:

```
>>> sequencetools.partition_sequence_by_weights_at_most(
...     sequence, [10, 5], cyclic=True, overhang=True)
[[3, 3, 3], [3], [4, 4], [4], [4, 5], [5]]
```

Return list of sequence objects.

`sequencetools.partition_sequence_by_weights_exactly`

`sequencetools.partition_sequence_by_weights_exactly` (*sequence, weights, cyclic=False, overhang=False*)

New in version 1.1. Partition *sequence* by *weights* exactly.

```
>>> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5]
```

Example 1. Partition sequence once by weights exactly without overhang:

```
>>> sequencetools.partition_sequence_by_weights_exactly(
...     sequence, [3, 9], cyclic=False, overhang=False)
[[3], [3, 3, 3]]
```

Example 2. Partition sequence once by weights exactly with overhang:

```
>>> sequencetools.partition_sequence_by_weights_exactly(
...     sequence, [3, 9], cyclic=False, overhang=True)
[[3], [3, 3, 3], [4, 4, 4, 4, 5]]
```

Example 3. Partition sequence cyclically by weights exactly without overhang:

```
>>> sequencetools.partition_sequence_by_weights_exactly(
...     sequence, [12], cyclic=True, overhang=False)
[[3, 3, 3, 3], [4, 4, 4]]
```


Example 4. Partition sequence cyclically by weights exactly with overhang:

```
>>> sequencetools.partition_sequence_by_weights_exactly(
...     sequence, [12], cyclic=True, overhang=True)
[[3, 3, 3, 3], [4, 4, 4], [4, 5]]
```

Return list sequence objects.

sequencetools.partition_sequence_extended_to_counts

`sequencetools.partition_sequence_extended_to_counts(sequence, counts, overhang=True)`

New in version 2.0. Partition sequence extended to counts.

Example 1. Partition sequence extended to counts with overhang:

```
>>> sequencetools.partition_sequence_extended_to_counts(
...     [1, 2, 3, 4], [6, 6, 6], overhang=True)
[[1, 2, 3, 4, 1, 2], [3, 4, 1, 2, 3, 4], [1, 2, 3, 4, 1, 2], [3, 4]]
```

Example 2. Partition sequence extended to counts without overhang:

```
>>> sequencetools.partition_sequence_extended_to_counts(
...     [1, 2, 3, 4], [6, 6, 6], overhang=False)
[[1, 2, 3, 4, 1, 2], [3, 4, 1, 2, 3, 4], [1, 2, 3, 4, 1, 2]]
```

Return sequence of sequence objects.

sequencetools.permute_sequence

`sequencetools.permute_sequence(sequence, permutation)`

New in version 2.0. Permute *sequence* by *permutation*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.permute_sequence([10, 11, 12, 13, 14, 15], [5, 4, 0, 1, 2, 3])
[15, 14, 10, 11, 12, 13]
```

Return newly constructed *sequence* object.

sequencetools.remove_sequence_elements_at_indices

`sequencetools.remove_sequence_elements_at_indices(sequence, indices)`

New in version 2.0. Remove *sequence* elements at *indices*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.remove_sequence_elements_at_indices(range(20), [1, 16, 17, 18])
[0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 19]
```

Ignore negative indices.

Return list.

sequencetools.remove_sequence_elements_at_indices_cyclically

`sequencetools.remove_sequence_elements_at_indices_cyclically(sequence, indices, period, offset=0)`

New in version 2.0. Remove *sequence* elements at *indices* mod *period* plus *offset*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.remove_sequence_elements_at_indices_cyclically(range(20), [0, 1], 5, 3)
[0, 1, 2, 5, 6, 7, 10, 11, 12, 15, 16, 17]
```

Ignore negative indices.

Return list.

sequencetools.remove_subsequence_of_weight_at_index

`sequencetools.remove_subsequence_of_weight_at_index(sequence, weight, index)`

New in version 1.1. Remove subsequence of *weight* at *index*:

```
>>> sequence = (1, 1, 2, 3, 5, 5, 1, 2, 5, 5, 6)

>>> sequencetools.remove_subsequence_of_weight_at_index(sequence, 13, 4)
(1, 1, 2, 3, 5, 5, 6)
```

Return newly constructed *sequence* object. Changed in version 2.0: renamed `listtools.remove_weighted_subrun_at()` to `sequencetools.remove_subsequence_of_weight_at_index()`.

sequencetools.repeat_runs_in_sequence_to_count

`sequencetools.repeat_runs_in_sequence_to_count(sequence, indicators)`

New in version 1.1. Repeat subruns in *sequence* according to *indicators*. The *indicators* input parameter must be a list of zero or more (start, length, count) triples. For every (start, length, count) indicator in *indicators*, the function copies `sequence[start:start+length]` and inserts count new copies of `sequence[start:start+length]` immediately after `sequence[start:start+length]` in *sequence*.

The function reads the value of count in every (start, length, count) triple not as the total number of occurrences of `sequence[start:start+length]` to appear in *sequence* after execution, but rather as the number of new occurrences of `sequence[start:start+length]` to appear in *sequence* after execution.

The function wraps newly created subruns in tuples. That is, this function returns output with one more level of nesting than given in input.

To insert 10 count of `sequence[:2]` at `sequence[2:2]`:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.repeat_runs_in_sequence_to_count(range(20), [(0, 2, 10)])
[0, 1, (0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1),
 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

To insert 5 count of `sequence[10:12]` at `sequence[12:12]` and then insert 5 count of `sequence[:2]` at `sequence[2:2]`:

```
>>> sequence = range(20)

>>> sequencetools.repeat_runs_in_sequence_to_count(sequence, [(0, 2, 5), (10, 2, 5)])
[0, 1, (0, 1, 0, 1, 0, 1, 0, 1, 0, 1), 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
(10, 11, 10, 11, 10, 11, 10, 11, 10, 11), 12, 13, 14, 15, 16, 17, 18, 19]
```

Note: This function wraps around the end of *sequence* whenever `len(sequence) < start + length`.

To insert 2 count of [18, 19, 0, 1] at `sequence[2:2]`:

```
>>> sequencetools.repeat_runs_in_sequence_to_count(sequence, [(18, 4, 2)])
[0, 1, (18, 19, 0, 1, 18, 19, 0, 1), 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19]
```

To insert 2 count of [18, 19, 0, 1, 2, 3, 4] at `sequence[4:4]`:

```
>>> sequencetools.repeat_runs_in_sequence_to_count(sequence, [(18, 8, 2)])
[0, 1, 2, 3, 4, 5, (18, 19, 0, 1, 2, 3, 4, 5, 18, 19, 0, 1, 2, 3, 4, 5), 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

Todo

Implement an optional *wrap* keyword to specify whether this function should wrap around the end of *sequence* whenever `len(sequence) < start + length` or not.

Todo

Reimplement this function to return a generator.

Generalizations of this function would include functions to repeat subruns in *sequence* to not only a certain count, as implemented here, but to a certain length, weight or sum. That is, `sequencetools.repeat_subruns_to_length()`, `sequencetools.repeat_subruns_to_weight()` and `sequencetools.repeat_subruns_to_sum()`. Changed in version 2.0: renamed `sequencetools.repeat_subruns_to_count()` to `sequencetools.repeat_runs_in_sequence_to_count()`.

sequencetools.repeat_sequence_elements_at_indices

`sequencetools.repeat_sequence_elements_at_indices(sequence, indices, total)`

New in version 2.0. Repeat *sequence* elements at *indices* to *total* length:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.repeat_sequence_elements_at_indices(range(10), [6, 7, 8], 3)
[0, 1, 2, 3, 4, 5, [6, 6, 6], [7, 7, 7], [8, 8, 8], 9]
```

Return list.

sequencetools.repeat_sequence_elements_at_indices_cyclically

`sequencetools.repeat_sequence_elements_at_indices_cyclically(sequence, cycle_token, total)`

New in version 2.0. Repeat *sequence* elements at indices specified by *cycle_token* to *total* length:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.repeat_sequence_elements_at_indices_cyclically(range(10), (5, [1, 2]), 3)
[0, [1, 1, 1], [2, 2, 2], 3, 4, 5, [6, 6, 6], [7, 7, 7], 8, 9]
```

The *cycle_token* may be a sieve:

```
>>> from abjad.tools import sievetools

>>> sieve = sievetools.cycle_tokens_to_sieve((5, [1, 2]))
>>> sequencetools.repeat_sequence_elements_at_indices_cyclically(range(10), sieve, 3)
[0, [1, 1, 1], [2, 2, 2], 3, 4, 5, [6, 6, 6], [7, 7, 7], 8, 9]
```

Return list.

sequencetools.repeat_sequence_elements_n_times_each

`sequencetools.repeat_sequence_elements_n_times_each(sequence, n)`

New in version 1.1. Repeat *sequence* elements *n* times each:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.repeat_sequence_elements_n_times_each((1, -1, 2, -3, 5, -5, 6), 2)
(1, 1, -1, -1, 2, 2, -3, -3, 5, 5, -5, -5, 6, 6)
```

Return newly constructed *sequence* object with copied *sequence* elements. Changed in version 2.0: renamed `listtools.repeat_elements_to_count()` to `sequencetools.repeat_sequence_elements_n_times_each()`.

sequencetools.repeat_sequence_n_times

`sequencetools.repeat_sequence_n_times(sequence, n)`

New in version 2.0. Repeat *sequence* *n* times:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.repeat_sequence_n_times((1, 2, 3, 4, 5), 3)
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
```

Repeat *sequence* 0 times:

```
>>> sequencetools.repeat_sequence_n_times((1, 2, 3, 4, 5), 0)
()
```

Return newly constructed *sequence* object of copied *sequence* elements.

sequencetools.repeat_sequence_to_length

`sequencetools.repeat_sequence_to_length(sequence, length, start=0)`

New in version 1.1. Repeat *sequence* to nonnegative integer *length*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.repeat_sequence_to_length(range(5), 11)
[0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0]
```

Repeat *sequence* to nonnegative integer *length* from *start*:

```
>>> sequencetools.repeat_sequence_to_length(range(5), 11, start=2)
[2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2]
```

Return newly constructed *sequence* object. Changed in version 2.0: renamed `listtools.repeat_list_to_length()` to `sequencetools.repeat_sequence_to_length()`.

sequencetools.repeat_sequence_to_weight_at_least

`sequencetools.repeat_sequence_to_weight_at_least(sequence, weight)`

New in version 1.1. Repeat *sequence* to *weight* at least:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.repeat_sequence_to_weight_at_least((5, -5, -5), 23)
(5, -5, -5, 5, -5)
```

Return newly constructed *sequence* object.

sequencetools.repeat_sequence_to_weight_at_most

`sequencetools.repeat_sequence_to_weight_at_most(sequence, weight)`

New in version 1.1. Repeat *sequence* to *weight* at most:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.repeat_sequence_to_weight_at_most((5, -5, -5), 23)
(5, -5, -5, 5)
```

Return newly constructed *sequence* object.

sequencetools.repeat_sequence_to_weight_exactly

`sequencetools.repeat_sequence_to_weight_exactly(sequence, weight)`

New in version 1.1. Repeat *sequence* to *weight* exactly:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.repeat_sequence_to_weight_exactly((5, -5, -5), 23)
(5, -5, -5, 5, -3)
```

Return newly constructed *sequence* object.

sequencetools.replace_sequence_elements_cyclically_with_new_material

`sequencetools.replace_sequence_elements_cyclically_with_new_material(sequence, indices, new_material)`

New in version 1.1. Replace *sequence* elements cyclically at *indices* with *new_material*:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.replace_sequence_elements_cyclically_with_new_material(
... range(20), ([0], 2), (['A', 'B'], 3))
['A', 1, 'B', 3, 4, 5, 'A', 7, 'B', 9, 10, 11, 'A', 13, 'B', 15, 16, 17, 'A', 19]

>>> sequencetools.replace_sequence_elements_cyclically_with_new_material(
... range(20), ([0], 2), (['*'], 1))
['*', 1, '*', 3, '*', 5, '*', 7, '*', 9, '*', 11, '*', 13, '*', 15, '*', 17, '*', 19]

>>> sequencetools.replace_sequence_elements_cyclically_with_new_material(
... range(20), ([0], 2), (['A', 'B', 'C', 'D'], None))
['A', 1, 'B', 3, 'C', 5, 'D', 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

>>> sequencetools.replace_sequence_elements_cyclically_with_new_material(
... range(20), ([0, 1, 8, 13], None), (['A', 'B', 'C', 'D'], None))
['A', 'B', 2, 3, 4, 5, 6, 7, 'C', 9, 10, 11, 12, 'D', 14, 15, 16, 17, 18, 19]
```

Raise type error when *sequence* not a list.

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.replace_elements_cyclic()` to `sequencetools.replace_sequence_elements_cyclically()`

sequencetools.retain_sequence_elements_at_indices

`sequencetools.retain_sequence_elements_at_indices(sequence, indices)`

New in version 2.0. Retain *sequence* elements at *indices*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.retain_sequence_elements_at_indices(range(20), [1, 16, 17, 18])
[1, 16, 17, 18]
```

Return sequence elements in the order they appear in *sequence*.

Ignore negative indices.

Return list.

sequencetools.retain_sequence_elements_at_indices_cyclically

`sequencetools.retain_sequence_elements_at_indices_cyclically(sequence, indices, period, offset=0)`

New in version 2.0. Retain *sequence* elements at *indices* mod *period* plus *offset*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.retain_sequence_elements_at_indices_cyclically(range(20), [0, 1], 5, 3)
[3, 4, 8, 9, 13, 14, 18, 19]
```

Ignore negative values in *indices*.

Return list.

sequencetools.reverse_sequence

`sequencetools.reverse_sequence(sequence)`

New in version 2.0. Reverse *sequence*:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.reverse_sequence((1, 2, 3, 4, 5))
(5, 4, 3, 2, 1)
```

Return new *sequence* object.

sequencetools.reverse_sequence_elements

`sequencetools.reverse_sequence_elements(sequence)`
 New in version 2.0. Reverse *sequence* elements:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.reverse_sequence_elements([1, (2, 3, 4), 5, (6, 7)])
[1, (4, 3, 2), 5, (7, 6)]
```

Return new *sequence* object.

sequencetools.rotate_sequence

`sequencetools.rotate_sequence(sequence, n)`
 New in version 1.1. Rotate *sequence* to the right:

```
>>> from abjad.tools import sequencetools

>>> sequencetools.rotate_sequence(range(10), 4)
[6, 7, 8, 9, 0, 1, 2, 3, 4, 5]
```

Rotate *sequence* to the left:

```
>>> sequencetools.rotate_sequence(range(10), -3)
[3, 4, 5, 6, 7, 8, 9, 0, 1, 2]
```

Rotate *sequence* neither to the right nor the left:

```
>>> sequencetools.rotate_sequence(range(10), 0)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Return newly created *sequence* object. Changed in version 2.0: renamed `sequencetools.rotate()` to `sequencetools.rotate_sequence()`.

sequencetools.splice_new_elements_between_sequence_elements

`sequencetools.splice_new_elements_between_sequence_elements(sequence, new_elements, overhang=(0, 0))`
 New in version 1.1. Splice copies of *new_elements* between each of the elements of *sequence*:

```
>>> from abjad.tools import sequencetools

>>> sequence = [0, 1, 2, 3, 4]
>>> new_elements = ['A', 'B']
```

```
>>> sequencetools.splice_new_elements_between_sequence_elements(sequence, new_elements)
[0, 'A', 'B', 1, 'A', 'B', 2, 'A', 'B', 3, 'A', 'B', 4]
```

Splice copies of *new_elements* between each of the elements of *sequence* and after the last element of *sequence*:

```
>>> sequencetools.splice_new_elements_between_sequence_elements(
...     sequence, new_elements, overhang=(0, 1))
[0, 'A', 'B', 1, 'A', 'B', 2, 'A', 'B', 3, 'A', 'B', 4, 'A', 'B']
```

Splice copies of *new_elements* before the first element of *sequence* and between each of the other elements of *sequence*:

```
>>> sequencetools.splice_new_elements_between_sequence_elements(
...     sequence, new_elements, overhang=(1, 0))
['A', 'B', 0, 'A', 'B', 1, 'A', 'B', 2, 'A', 'B', 3, 'A', 'B', 4]
```

Splice copies of *new_elements* before the first element of *sequence*, after the last element of *sequence* and between each of the other elements of *sequence*:

```
>>> sequencetools.splice_new_elements_between_sequence_elements(
...     sequence, new_elements, overhang=(1, 1))
['A', 'B', 0, 'A', 'B', 1, 'A', 'B', 2, 'A', 'B', 3, 'A', 'B', 4, 'A', 'B']
```

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.insert_slice_cyclic()` to `sequencetools.splice_new_elements_between_sequence_elements()`

sequencetools.split_sequence_by_weights

`sequencetools.split_sequence_by_weights` (*sequence*, *weights*, *cyclic=False*, *overhang=False*)
 New in version 2.0. Split sequence by weights.

Example 1. Split sequence cyclically by weights with overhang:

```
>>> sequencetools.split_sequence_by_weights(
...     (10, -10, 10, -10), [3, 15, 3], cyclic=True, overhang=True)
[(3,), (7, -8), (-2, 1), (3,), (6, -9), (-1,)]
```

Example 2. Split sequence cyclically by weights without overhang:

```
>>> sequencetools.split_sequence_by_weights(
...     (10, -10, 10, -10), [3, 15, 3], cyclic=True, overhang=False)
[(3,), (7, -8), (-2, 1), (3,), (6, -9)]
```

Example 3. Split sequence once by weights with overhang:

```
>>> sequencetools.split_sequence_by_weights(
...     (10, -10, 10, -10), [3, 15, 3], cyclic=False, overhang=True)
[(3,), (7, -8), (-2, 1), (9, -10)]
```

Example 4. Split sequence once by weights without overhang:

```
>>> sequencetools.split_sequence_by_weights(
...     (10, -10, 10, -10), [3, 15, 3], cyclic=False, overhang=False)
[(3,), (7, -8), (-2, 1)]
```

Return list of sequence types.

sequencetools.split_sequence_extended_to_weights

`sequencetools.split_sequence_extended_to_weights` (*sequence*, *weights*, *overhang=True*)

New in version 2.0. Split sequence extended to weights.

Example 1. Split sequence extended to weights with overhang:

```
>>> sequencetools.split_sequence_extended_to_weights(
...     [1, 2, 3, 4, 5], [7, 7, 7], overhang=True)
[[1, 2, 3, 1], [3, 4], [1, 1, 2, 3], [4, 5]]
```

Example 2. Split sequence extended to weights without overhang:

```
>>> sequencetools.split_sequence_extended_to_weights(
...     [1, 2, 3, 4, 5], [7, 7, 7], overhang=False)
[[1, 2, 3, 1], [3, 4], [1, 1, 2, 3]]
```

Return sequence of sequence objects.

sequencetools.sum_consecutive_sequence_elements_by_sign

`sequencetools.sum_consecutive_sequence_elements_by_sign` (*sequence*, *sign*=[-1, 0, 1])

New in version 1.1. Sum consecutive *sequence* elements by *sign*:

```
>>> from abjad.tools import sequencetools

>>> sequence = [0, 0, -1, -1, 2, 3, -5, 1, 2, 5, -5, -6]

>>> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence)
[0, -2, 5, -5, 8, -11]

>>> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign=[-1])
[0, 0, -2, 2, 3, -5, 1, 2, 5, -11]

>>> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign=[0])
[0, -1, -1, 2, 3, -5, 1, 2, 5, -5, -6]

>>> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign=[1])
[0, 0, -1, -1, 5, -5, 8, -5, -6]

>>> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign=[-1, 0])
[0, -2, 2, 3, -5, 1, 2, 5, -11]

>>> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign=[-1, 1])
[0, 0, -2, 5, -5, 8, -11]

>>> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign=[0, 1])
[0, -1, -1, 5, -5, 8, -5, -6]

>>> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign=[-1, 0, 1])
[0, -2, 5, -5, 8, -11]
```

When -1 in *sign*, sum consecutive negative elements.

When 0 in *sign*, sum consecutive 0 elements.

When 1 in *sign*, sum consecutive positive elements.

Return list. Changed in version 2.0: renamed `sequencetools.sum_by_sign()` to `sequencetools.sum_consecutive_sequence_elements_by_sign()`.

`sequencetools.sum_sequence_elements_at_indices`

`sequencetools.sum_sequence_elements_at_indices(sequence, pairs, period=None, overhang=True)`

New in version 1.1. Sum *sequence* elements at indices according to *pairs*:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.sum_sequence_elements_at_indices(range(10), [(0, 3)])
[3, 3, 4, 5, 6, 7, 8, 9]
```

Sum *sequence* elements cyclically at indices according to *pairs* and *period*:

```
>>> sequencetools.sum_sequence_elements_at_indices(range(10), [(0, 3)], period=4)
[3, 3, 15, 7, 17]
```

Sum *sequence* elements cyclically at indices according to *pairs* and *period* and do not return incomplete final sum:

```
>>> sequencetools.sum_sequence_elements_at_indices(
...     range(10), [(0, 3)], period=4, overhang=False)
[3, 3, 15, 7]
```

Replace `sequence[i:i+count]` with `sum(sequence[i:i+count])` for each `(i, count)` in *pairs*.

Indices in *pairs* must be less than *period* when *period* is not none.

Return new list. Changed in version 2.0: renamed `sequencetools.sum_slices_at()` to `sequencetools.sum_sequence_elements_at_indices()`.

`sequencetools.truncate_runs_in_sequence`

`sequencetools.truncate_runs_in_sequence(sequence)`

New in version 1.1. Truncate subruns of like elements in *sequence* to length 1:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.truncate_runs_in_sequence([1, 1, 2, 3, 3, 3, 9, 4, 4, 4])
[1, 2, 3, 9, 4]
```

Return empty list when *sequence* is empty:

```
>>> sequencetools.truncate_runs_in_sequence([])
[]
```

Raise type error when *sequence* is not a list.

Return new list. Changed in version 2.0: renamed `sequencetools.truncate_subruns()` to `sequencetools.truncate_runs_in_sequence()`.

`sequencetools.truncate_sequence_to_sum`

`sequencetools.truncate_sequence_to_sum(sequence, target_sum)`

New in version 1.1. Truncate *sequence* to *target_sum*:

```
>>> sequence = [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]

>>> for n in range(10):
...     print n, sequencetools.truncate_sequence_to_sum(sequence, n)
...
0 []
1 [-1, 2]
2 [-1, 2, -3, 4]
3 [-1, 2, -3, 4, -5, 6]
4 [-1, 2, -3, 4, -5, 6, -7, 8]
5 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
6 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
7 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
8 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
9 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
```

Return empty list when *target_sum* is 0:

```
>>> sequencetools.truncate_sequence_to_sum([1, 2, 3, 4, 5], 0)
[]
```

Raise type error when *sequence* is not a list.

Raise value error on negative *target_sum*.

Return new list. Changed in version 2.0: renamed `sequencetools.truncate_to_sum()` to `sequencetools.truncate_sequence_to_sum()`.

sequencetools.truncate_sequence_to_weight

`sequencetools.truncate_sequence_to_weight(sequence, weight)`

New in version 1.1. Truncate *sequence* to *weight*:

```
>>> from abjad.tools import sequencetools

>>> l = [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
>>> for x in range(10):
...     print x, sequencetools.truncate_sequence_to_weight(l, x)
...
0 []
1 [-1]
2 [-1, 1]
3 [-1, 2]
4 [-1, 2, -1]
5 [-1, 2, -2]
6 [-1, 2, -3]
7 [-1, 2, -3, 1]
8 [-1, 2, -3, 2]
9 [-1, 2, -3, 3]
```

Return empty list when *weight* is 0:

```
>>> sequencetools.truncate_sequence_to_weight([1, 2, 3, 4, 5], 0)
[]
```

Raise type error when *sequence* is not a list.

Raise value error on negative *weight*.

Return new list. Changed in version 2.0: renamed `sequencetools.truncate_to_weight()` to `sequencetools.truncate_sequence_to_weight()`.

`sequencetools.yield_all_combinations_of_sequence_elements`

`sequencetools.yield_all_combinations_of_sequence_elements` (*sequence*,
min_length=None,
max_length=None)

New in version 2.0. Yield all combinations of *sequence* in binary string order:

```
>>> list(sequencetools.yield_all_combinations_of_sequence_elements(
...     [1, 2, 3, 4]))
[[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3], [4], [1, 4],
 [2, 4], [1, 2, 4], [3, 4], [1, 3, 4], [2, 3, 4], [1, 2, 3, 4]]
```

Yield all combinations of *sequence* greater than or equal to *min_length* in binary string order:

```
>>> list(sequencetools.yield_all_combinations_of_sequence_elements(
...     [1, 2, 3, 4], min_length=3))
[[1, 2, 3], [1, 2, 4], [1, 3, 4], [2, 3, 4], [1, 2, 3, 4]]
```

Yield all combinations of *sequence* less than or equal to *max_length* in binary string order:

```
>>> list(sequencetools.yield_all_combinations_of_sequence_elements(
...     [1, 2, 3, 4], max_length=2))
[[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [4], [1, 4], [2, 4], [3, 4]]
```

Yield all combinations of *sequence* greater than or equal to *min_length* and less than or equal to *max_length* in lex order:

```
>>> list(sequencetools.yield_all_combinations_of_sequence_elements(
...     [1, 2, 3, 4], min_length=2, max_length=2))
[[1, 2], [1, 3], [2, 3], [1, 4], [2, 4], [3, 4]]
```

Return generator of newly created *sequence* objects. Changed in version 2.0: renamed `sequencetools.sublists()` to `sequencetools.yield_all_combinations_of_sequence_elements()`.

`sequencetools.yield_all_k_ary_sequences_of_length`

`sequencetools.yield_all_k_ary_sequences_of_length` (*k*, *length*)

New in version 2.0. Generate all *k*-ary sequences of *length*:

```
>>> from abjad.tools import sequencetools

>>> for sequence in sequencetools.yield_all_k_ary_sequences_of_length(2, 3):
...     sequence
...
(0, 0, 0)
(0, 0, 1)
(0, 1, 0)
(0, 1, 1)
(1, 0, 0)
(1, 0, 1)
(1, 1, 0)
(1, 1, 1)
```

Return generator of tuples.

sequencetools.yield_all_pairs_between_sequences

`sequencetools.yield_all_pairs_between_sequences` (*l*, *m*)

New in version 2.0. Yield all pairs between sequences *l* and *m*:

```
>>> from abjad.tools import sequencetools

>>> for pair in sequencetools.yield_all_pairs_between_sequences([1, 2, 3], [4, 5]):
...     pair
...
(1, 4)
(1, 5)
(2, 4)
(2, 5)
(3, 4)
(3, 5)
```

Return pair generator.

sequencetools.yield_all_partitions_of_sequence

`sequencetools.yield_all_partitions_of_sequence` (*sequence*)

New in version 2.0. Yield all partitions of *sequence*:

```
>>> from abjad.tools import sequencetools

>>> for partition in sequencetools.yield_all_partitions_of_sequence([0, 1, 2, 3]):
...     partition
...
[[0, 1, 2, 3]]
[[0, 1, 2], [3]]
[[0, 1], [2, 3]]
[[0, 1], [2], [3]]
[[0], [1, 2, 3]]
[[0], [1, 2], [3]]
[[0], [1], [2, 3]]
[[0], [1], [2], [3]]
```

Return generator of newly created lists.

sequencetools.yield_all_permutations_of_sequence

`sequencetools.yield_all_permutations_of_sequence` (*sequence*)

New in version 1.1. Yield all permutations of *sequence* in lex order:

```
>>> from abjad.tools import sequencetools

>>> list(sequencetools.yield_all_permutations_of_sequence((1, 2, 3)))
[(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```

Return generator of *sequence* objects. Changed in version 2.0: renamed `listtools.permutations()` to `sequencetools.yield_all_permutations_of_sequence()`.

`sequencetools.yield_all_permutations_of_sequence_in_orbit`

`sequencetools.yield_all_permutations_of_sequence_in_orbit` (*sequence*, *permutation*)

New in version 2.0. Yield all permutations of *sequence* in orbit of *permutation* in lex order:

```
>>> list(sequencetools.yield_all_permutations_of_sequence_in_orbit(
...      (1, 2, 3, 4), [1, 2, 3, 0]))
[(1, 2, 3, 4), (2, 3, 4, 1), (3, 4, 1, 2), (4, 1, 2, 3)]
```

Return generator of *sequence* objects.

`sequencetools.yield_all_restricted_growth_functions_of_length`

`sequencetools.yield_all_restricted_growth_functions_of_length` (*length*)

New in version 2.0. Generate all restricted growth functions of *length* in lex order:

```
>>> from abjad.tools import sequencetools

>>> for rgf in sequencetools.yield_all_restricted_growth_functions_of_length(4):
...     rgf
...
(1, 1, 1, 1)
(1, 1, 1, 2)
(1, 1, 2, 1)
(1, 1, 2, 2)
(1, 1, 2, 3)
(1, 2, 1, 1)
(1, 2, 1, 2)
(1, 2, 1, 3)
(1, 2, 2, 1)
(1, 2, 2, 2)
(1, 2, 2, 3)
(1, 2, 3, 1)
(1, 2, 3, 2)
(1, 2, 3, 3)
(1, 2, 3, 4)
```

Return generator of tuples.

`sequencetools.yield_all_rotations_of_sequence`

`sequencetools.yield_all_rotations_of_sequence` (*sequence*, *n=1*)

New in version 2.0. Yield all *n*-rotations of *sequence* up to identity:

```
>>> from abjad.tools import sequencetools

>>> list(sequencetools.yield_all_rotations_of_sequence([1, 2, 3, 4], -1))
[[1, 2, 3, 4], [2, 3, 4, 1], [3, 4, 1, 2], [4, 1, 2, 3]]
```

Return generator of *sequence* objects.

`sequencetools.yield_all_set_partitions_of_sequence`

`sequencetools.yield_all_set_partitions_of_sequence` (*sequence*)

New in version 2.0. Yield all set partitions of *sequence* in restricted growth function order:

```
>>> for set_partition in sequencetools.yield_all_set_partitions_of_sequence(
...     [21, 22, 23, 24]):
...     set_partition
...
[[21, 22, 23, 24]]
[[21, 22, 23], [24]]
[[21, 22, 24], [23]]
[[21, 22], [23, 24]]
[[21, 22], [23], [24]]
[[21, 23, 24], [22]]
[[21, 23], [22, 24]]
[[21, 23], [22], [24]]
[[21, 24], [22, 23]]
[[21], [22, 23, 24]]
[[21], [22, 23], [24]]
[[21, 24], [22], [23]]
[[21], [22, 24], [23]]
[[21], [22], [23, 24]]
[[21], [22], [23], [24]]
```

Return generator of list of lists.

sequencetools.yield_all_subsequences_of_sequence

`sequencetools.yield_all_subsequences_of_sequence(sequence, min_length=0, max_length=None)`

New in version 2.0. Yield all subsequences of *sequence* in lex order:

```
>>> from abjad.tools import sequencetools

>>> list(sequencetools.yield_all_subsequences_of_sequence([0, 1, 2]))
[[], [0], [0, 1], [0, 1, 2], [1], [1, 2], [2]]
```

Yield all subsequences of *sequence* greater than or equal to *min_length* in lex order:

```
>>> list(sequencetools.yield_all_subsequences_of_sequence(
...     [0, 1, 2, 3, 4], min_length=3))
[[0, 1, 2], [0, 1, 2, 3], [0, 1, 2, 3, 4], [1, 2, 3], [1, 2, 3, 4], [2, 3, 4]]
```

Yield all subsequences of *sequence* less than or equal to *max_length* in lex order:

```
>>> for subsequence in sequencetools.yield_all_subsequences_of_sequence(
...     [0, 1, 2, 3, 4], max_length=3):
...     subsequence
...
[]
[0]
[0, 1]
[0, 1, 2]
[1]
[1, 2]
[1, 2, 3]
[2]
[2, 3]
[2, 3, 4]
[3]
[3, 4]
[4]
```

Yield all subsequences of *sequence* greater than or equal to *min_length* and less than or equal to *max_length* in lex order:

```
>>> list(sequencetools.yield_all_subsequences_of_sequence(
...     [0, 1, 2, 3, 4], min_length=3, max_length=3))
[[0, 1, 2], [1, 2, 3], [2, 3, 4]]
```

Return generator of newly created *sequence* slices.

sequencetools.yield_all_unordered_pairs_of_sequence

sequencetools.yield_all_unordered_pairs_of_sequence(*sequence*)

New in version 2.0. Yield all unordered pairs of *sequence*:

```
>>> from abjad.tools import sequencetools

>>> list(sequencetools.yield_all_unordered_pairs_of_sequence([1, 2, 3, 4]))
[(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
```

Yield all unordered pairs of length-1 *sequence*:

```
>>> list(sequencetools.yield_all_unordered_pairs_of_sequence([1]))
[]
```

Yield all unordered pairs of empty *sequence*:

```
>>> list(sequencetools.yield_all_unordered_pairs_of_sequence([]))
[]
```

Yield all unordered pairs of *sequence* with duplicate elements:

```
>>> list(sequencetools.yield_all_unordered_pairs_of_sequence([1, 1, 1]))
[(1, 1), (1, 1), (1, 1)]
```

Pairs are tuples instead of sets to accommodate duplicate *sequence* elements.

Return generator.

sequencetools.yield_outer_product_of_sequences

sequencetools.yield_outer_product_of_sequences(*sequences*)

New in version 1.1. Yield outer product of *sequences*:

```
>>> list(sequencetools.yield_outer_product_of_sequences(
...     [[1, 2, 3], ['a', 'b']]))
[[1, 'a'], [1, 'b'], [2, 'a'], [2, 'b'], [3, 'a'], [3, 'b']]

>>> list(sequencetools.yield_outer_product_of_sequences(
...     [[1, 2, 3], ['a', 'b'], ['X', 'Y']]))
[[1, 'a', 'X'], [1, 'a', 'Y'], [1, 'b', 'X'], [1, 'b', 'Y'],
 [2, 'a', 'X'], [2, 'a', 'Y'], [2, 'b', 'X'], [2, 'b', 'Y'],
 [3, 'a', 'X'], [3, 'a', 'Y'], [3, 'b', 'X'], [3, 'b', 'Y']]

>>> list(sequencetools.yield_outer_product_of_sequences(
...     [[1, 2, 3], [4, 5], [6, 7, 8]]))
[[1, 4, 6], [1, 4, 7], [1, 4, 8], [1, 5, 6], [1, 5, 7], [1, 5, 8],
 [2, 4, 6], [2, 4, 7], [2, 4, 8], [2, 5, 6], [2, 5, 7], [2, 5, 8],
 [3, 4, 6], [3, 4, 7], [3, 4, 8], [3, 5, 6], [3, 5, 7], [3, 5, 8]]
```


Return generator. Changed in version 2.0: renamed `sequencetools.outer_product()` to `sequencetools.yield_outer_product_of_sequences()`.

sequencetools.zip_sequences_cyclically

`sequencetools.zip_sequences_cyclically(*sequences)`

New in version 1.1. Zip *sequences* cyclically:

```
>>> from abjad.tools import sequencetools
```

```
>>> sequencetools.zip_sequences_cyclically([1, 2, 3], ['a', 'b'])
[(1, 'a'), (2, 'b'), (3, 'a')]
```

New in version 1.1: Arbitrary number of input sequences now allowed.

```
>>> sequencetools.zip_sequences_cyclically([10, 11, 12], [20, 21], [30, 31, 32, 33])
[(10, 20, 30), (11, 21, 31), (12, 20, 32), (10, 21, 33)]
```

Cycle over the elements of the sequences of shorter length.

Return list of length equal to sequence of greatest length in *sequences*. Changed in version 2.0: renamed `sequencetools.zip_cyclic()` to `sequencetools.zip_sequences_cyclically()`.

sequencetools.zip_sequences_without_truncation

`sequencetools.zip_sequences_without_truncation(*sequences)`

New in version 1.1. Zip *sequences* nontruncating:

```
>>> sequencetools.zip_sequences_without_truncation(
...     [1, 2, 3, 4], [11, 12, 13], [21, 22, 23])
[(1, 11, 21), (2, 12, 22), (3, 13, 23), (4,)]
```

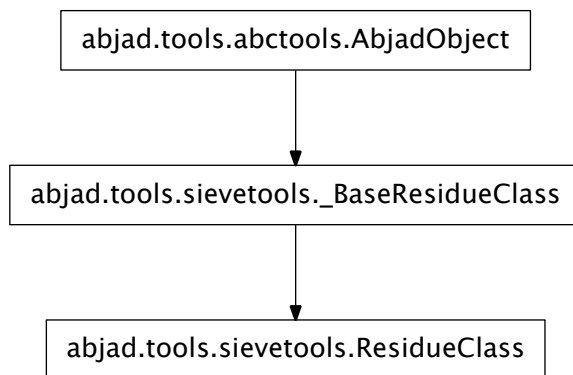
Lengths of the tuples returned may differ but will always be greater than or equal to 1.

Return list of tuples. Changed in version 2.0: renamed `sequencetools.zip_nontruncating()` to `sequencetools.zip_sequences_without_truncation()`.

sievetools

concrete classes

sievetools.ResidueClass



class sievetools.**ResidueClass** (*args)

Residue class (or congruence class). Residue classes form the basis of Xenakis sieves. They can be used to construct any complex periodic integer (or boolean) sequence as a combination of simple periodic sequences.

Example from the opening of Xenakis’s *Psappha* for solo percussion:

```

>>> from abjad.tools.sievetools import ResidueClass as RC

>>> s1 = (RC(8, 0) | RC(8, 1) | RC(8, 7)) & (RC(5, 1) | RC(5, 3))
>>> s2 = (RC(8, 0) | RC(8, 1) | RC(8, 2)) & RC(5, 0)
>>> s3 = RC(8, 3)
>>> s4 = RC(8, 4)
>>> s5 = (RC(8, 5) | RC(8, 6)) & (RC(5, 2) | RC(5, 3) | RC(5, 4))
>>> s6 = (RC(8, 1) & RC(5, 2))
>>> s7 = (RC(8, 6) & RC(5, 1))

>>> y = s1 | s2 | s3 | s4 | s5 | s6 | s7

>>> y.get_congruent_bases(40)
[0, 1, 3, 4, 6, 8, 10, 11, 12, 13, 14, 16, 17, 19, 20, 22, 23, 25, 27,
 28, 29, 31, 33, 35, 36, 37, 38, 40]

>>> y.get_boolean_train(40)
[1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0]
  
```

Return residue class.

Read-only Properties

`ResidueClass.modulo`
 Period of residue class.

`ResidueClass.residue`
 Residue of residue class.

`ResidueClass.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`ResidueClass.get_boolean_train(*min_max)`
 Returns a boolean train with 0s mapped to the integers that are not congruent bases of the residue class and 1s mapped to those that are. The method takes one or two integer arguments. If only one is given, it is taken as the max range and the min is assumed to be 0.

Example:

```
>>> from abjad.tools.sievetools import ResidueClass as RC

>>> r = RC(3, 0)
>>> r.get_boolean_train(6)
[1, 0, 0, 1, 0, 0]
>>> r.get_congruent_bases(-6, 6)
[-6, -3, 0, 3, 6]
```

Return list.

`ResidueClass.get_congruent_bases(*min_max)`
 Returns all the congruent bases of this residue class within the given range. The method takes one or two integer arguments. If only one it given, it is taken as the max range and the min is assumed to be 0.

Example:

```
>>> from abjad.tools.sievetools import ResidueClass as RC

>>> r = RC(3, 0)
>>> r.get_congruent_bases(6)
[0, 3, 6]
>>> r.get_congruent_bases(-6, 6)
[-6, -3, 0, 3, 6]
```

Return list.

Special Methods

`ResidueClass.__and__(arg)`
 Inherited from `sievetools._BaseResidueClass`

`ResidueClass.__eq__(exp)`

`ResidueClass.__ge__(expr)`

`ResidueClass.__gt__(expr)`

`ResidueClass.__le__(expr)`

`ResidueClass.__lt__(expr)`

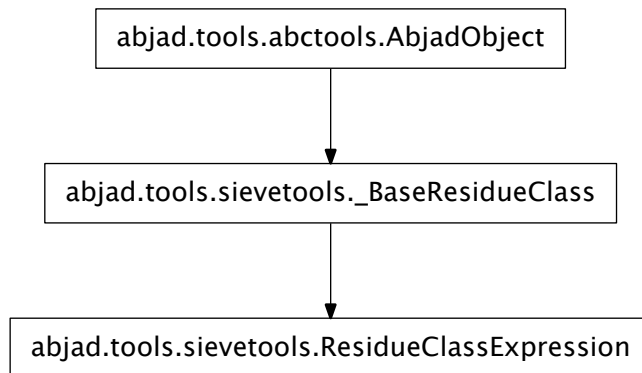
`ResidueClass.__ne__(expr)`

`ResidueClass.__or__(arg)`
 Inherited from `sievetools._BaseResidueClass`

`ResidueClass.__repr__()`

`ResidueClass.__xor__(arg)`
 Inherited from `sievetools._BaseResidueClass`

`sievetools.ResidueClassExpression`



class `sievetools.ResidueClassExpression` (*r**cs*, *operator*='or')

Read-only Properties

`ResidueClassExpression.operator`
 Operator of residue class expression.

`ResidueClassExpression.period`

`ResidueClassExpression.rcs`
 Residue classes of expression.

`ResidueClassExpression.representative_boolean_train`

`ResidueClassExpression.representative_congruent_bases`

`ResidueClassExpression.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`ResidueClassExpression.get_boolean_train(*min_max)`
 Returns a boolean train with 0s mapped to the integers that are not congruent bases of the RC expression and 1s mapped to those that are. The method takes one or two integer arguments. If only one is given, it is taken as the max range and min is assumed to be 0.

Example:

```
>>> from abjad.tools.sievetools import ResidueClass as RC

>>> e = RC(3, 0) | RC(2, 0)
>>> e.get_boolean_train(6)
[1, 0, 1, 1, 1, 0]
>>> e.get_congruent_bases(-6, 6)
[-6, -4, -3, -2, 0, 2, 3, 4, 6]
```

Return list.

`ResidueClassExpression.get_congruent_bases(*min_max)`

Returns all the congruent bases of this RC expression within the given range. The method takes one or two integer arguments. If only one it given, it is taken as the max range and min is assumed to be 0.

Example:

```
>>> from abjad.tools.sievetools import ResidueClass as RC

>>> e = RC(3, 0) | RC(2, 0)
>>> e.get_congruent_bases(6)
[0, 2, 3, 4, 6]
>>> e.get_congruent_bases(-6, 6)
[-6, -4, -3, -2, 0, 2, 3, 4, 6]
```

Return list.

`ResidueClassExpression.is_congruent_base(integer)`

Special Methods

`ResidueClassExpression.__and__(arg)`

Inherited from `sievetools._BaseResidueClass`

`ResidueClassExpression.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__or__(arg)`
 Inherited from `sievetools._BaseResidueClass`

`ResidueClassExpression.__repr__()`

`ResidueClassExpression.__xor__(arg)`
 Inherited from `sievetools._BaseResidueClass`

functions

`sievetools.all_are_residue_class_expressions`

`sievetools.all_are_residue_class_expressions(expr)`
 New in version 2.6. True when *expr* is a sequence of Abjad residue class expressions:

```
>>> from abjad.tools import sievetools

>>> sieve = sievetools.ResidueClass(3, 0) | sievetools.ResidueClass(2, 0)

>>> sieve
{ResidueClass(2, 0) | ResidueClass(3, 0)}

>>> sievetools.all_are_residue_class_expressions([sieve])
True
```

True when *expr* is an empty sequence:

```
>>> sievetools.all_are_residue_class_expressions([])
True
```

Otherwise false:

```
>>> sievetools.all_are_residue_class_expressions('foo')
False
```

Return boolean.

`sievetools.cycle_tokens_to_sieve`

`sievetools.cycle_tokens_to_sieve(*cycle_tokens)`
 New in version 2.0. Make Xenakis sieve from arbitrarily many *cycle_tokens*:

```
>>> from abjad.tools import sievetools

>>> cycle_token_1 = (6, [0, 4, 5])
>>> cycle_token_2 = (10, [0, 1, 2], 6)
>>> sievetools.cycle_tokens_to_sieve(cycle_token_1, cycle_token_2)
```

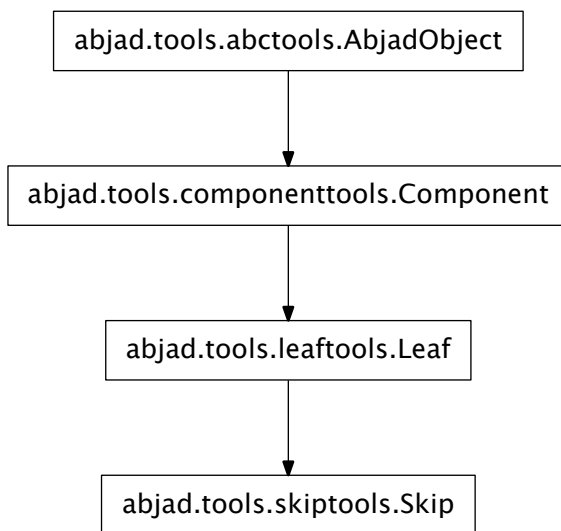
```
{ResidueClass(6, 0) | ResidueClass(6, 4) | ResidueClass(6, 5) |
 ResidueClass(10, 6) | ResidueClass(10, 7) | ResidueClass(10, 8)}
```

Cycle token comprises mandatory *modulo*, mandatory *residues* and optional *offset*.

skiptools

concrete classes

skiptools.Skip



```
class skiptools.Skip(*args, **kwargs)
    Abjad model of a LilyPond skip:
```

```
>>> skiptools.Skip((3, 16))
Skip('s8.')
```

Return skip.

Read-only Properties

Skip.duration_in_seconds

Inherited from `leaftools.Leaf`

Skip.leaf_index

Inherited from `leaftools.Leaf`

Skip.lilypond_format

Inherited from `componenttools.Component`

Skip.multiplied_duration

Inherited from `leaftools.Leaf`

Skip.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Skip.parent

Inherited from `componenttools.Component`

Skip.preprolated_duration

Inherited from `leaftools.Leaf`

Skip.prolated_duration

Inherited from `componenttools.Component`

Skip.prolation

Inherited from `componenttools.Component`

Skip.set

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Skip.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Skip.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

Skip.start_offset_in_seconds

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

Skip.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

Skip.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Skip.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Skip.duration_multiplier

Inherited from `leaftools.Leaf`

Skip.written_duration

Inherited from `leaftools.Leaf`

Skip.written_pitch_indication_is_at_sounding_pitch

Inherited from `leaftools.Leaf`

`Skip.written_pitch_indication_is_nonsemantic`
 Inherited from `leaftools.Leaf`

Special Methods

`Skip.__and__(arg)`

Inherited from `leaftools.Leaf`

`Skip.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Skip.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Skip.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Skip.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Skip.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Skip.__mul__(n)`

Inherited from `componenttools.Component`

`Skip.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Skip.__or__(arg)`

Inherited from `leaftools.Leaf`

`Skip.__repr__()`

Inherited from `leaftools.Leaf`

`Skip.__rmul__(n)`

Inherited from `componenttools.Component`

`Skip.__str__()`

Inherited from `leaftools.Leaf`

`Skip.__sub__(arg)`

Inherited from `leaftools.Leaf`

`Skip.__xor__(arg)`
Inherited from `leaftools.Leaf`

functions

`skiptools.all_are_skips`

`skiptools.all_are_skips(expr)`
New in version 2.6. True when *expr* is a sequence of Abjad skips:

```
>>> from abjad.tools import skiptools

>>> skips = 3 * skiptools.Skip('s4')

>>> skips
[Skip('s4'), Skip('s4'), Skip('s4')]

>>> skiptools.all_are_skips(skips)
True
```

True when *expr* is an empty sequence:

```
>>> skiptools.all_are_skips([])
True
```

Otherwise false:

```
>>> skiptools.all_are_skips('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

`skiptools.make_repeated_skips_from_time_signature`

`skiptools.make_repeated_skips_from_time_signature(time_signature)`
New in version 2.0. Make repeated skips from *time_signature*:

```
>>> skiptools.make_repeated_skips_from_time_signature((5, 32))
[Skip('s32'), Skip('s32'), Skip('s32'), Skip('s32'), Skip('s32')]
```

Return list of skips.

`skiptools.make_repeated_skips_from_time_signatures`

`skiptools.make_repeated_skips_from_time_signatures(time_signatures)`
Make repeated skips from *time_signatures*:

```
skiptools.make_repeated_skips_from_time_signatures([(2, 8), (3, 32)])
[[Skip('s8'), Skip('s8')], [Skip('s32'), Skip('s32'), Skip('s32')]]
```

Return list of skip lists.

skiptools.make_skips_with_multiplied_durations

`skiptools.make_skips_with_multiplied_durations(written_duration, multiplied_durations)`

New in version 2.0. Make *written_duration* skips with *multiplied_durations*:

```
>>> skiptools.make_skips_with_multiplied_durations(
...     Duration(1, 4), [(1, 2), (1, 3), (1, 4), (1, 5)])
[Skip('s4 * 2'), Skip('s4 * 4/3'), Skip('s4 * 1'), Skip('s4 * 4/5')]
```

Useful for making invisible layout voices.

Return list of skips. Changed in version 2.0: renamed `construct.skips_with_multipliers()` to `skiptools.make_skips_with_multiplied_durations()`.

skiptools.replace_leaves_in_expr_with_skips

`skiptools.replace_leaves_in_expr_with_skips(expr)`

New in version 1.1. Replace leaves in *expr* with skips:

```
>>> staff = Staff(Measure((2, 8), "c'8 d'8") * 2)
>>> skiptools.replace_leaves_in_expr_with_skips(staff[0])

>>> f(staff)
\new Staff {
  {
    \time 2/8
    s8
    s8
  }
  {
    c'8
    d'8
  }
}
```

Return none. Changed in version 2.0: renamed `leaftools.replace_leaves_with_skips_in()` to `skiptools.replace_leaves_in_expr_with_skips()`.

skiptools.yield_groups_of_skips_in_sequence

`skiptools.yield_groups_of_skips_in_sequence(sequence)`

New in version 2.0. Yield groups of skips in *sequence*:

```
>>> staff = Staff("c'8 d'8 s8 s8 <e' g'>8 <f' a'>8 g'8 a'8 s8 s8 <b' d''>8 <c'' e''>8")

>>> f(staff)
\new Staff {
  c'8
  d'8
  s8
  s8
  <e' g'>8
  <f' a'>8
  g'8
  a'8
  s8
}
```

```
s8
<b' d''>8
<c'' e''>8
}
```

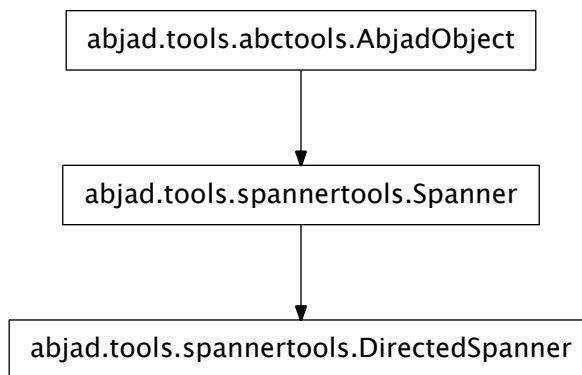
```
>>> for skip in skiptools.yield_groups_of_skips_in_sequence(staff):
...     skip
...
(Skip('s8'), Skip('s8'))
(Skip('s8'), Skip('s8'))
```

Return generator.

spannertools

abstract classes

spannertools.DirectedSpanner



```
class spannertools.DirectedSpanner(components=[], direction=None)
```

Read-only Properties

`DirectedSpanner.components`

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`DirectedSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`DirectedSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`DirectedSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`DirectedSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`DirectedSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`DirectedSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`DirectedSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`DirectedSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`DirectedSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`DirectedSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`DirectedSpanner.direction`

Methods

`DirectedSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DirectedSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DirectedSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`DirectedSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DirectedSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return `none`.

Inherited from `spannertools.Spanner`

`DirectedSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are `Left`, `Right` and `None`.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`DirectedSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
```

```

        f'8 ]
    }

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}

```

Return list.

Inherited from `spannertools.Spanner`

`DirectedSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0

```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`DirectedSpanner.pop()`

Remove and return rightmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)

```

Return component.

Inherited from `spannertools.Spanner`

`DirectedSpanner.pop_left()`

Remove and return leftmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

```



```
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`DirectedSpanner.__call__(expr)`

New in version 2.9. Call `spanner` on *expr* as a shortcut to extend `spanner`:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying `spanner` and mark attachment syntax.

Return `spanner`.

Inherited from `spannertools.Spanner`

`DirectedSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`DirectedSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DirectedSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DirectedSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`DirectedSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DirectedSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

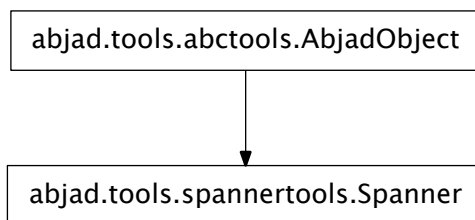
`DirectedSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`DirectedSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`DirectedSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`DirectedSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

`spannertools.Spanner`



class `spannertools.Spanner` (*components=None*)
 New in version 1.1. Any type of notation object that stretches horizontally and encompasses some number of notes, rest, chords, tuplets, measures, voices or other Abjad components.

Beams, slurs, hairpins, trills, glissandi and piano pedal brackets all stretch horizontally on the page to encompass multiple notes and all implement as Abjad spanners. That is, these spanner all have an obvious graphic reality with definite start-, stop- and midpoints.

Abjad also implements a number of spanners of a different type, such as tempo and instrument spanners, which mark a group of notes, rests, chords or measures as carrying a certain tempo or being played by a certain instrument.

The spanner class described here abstracts the functionality that all such spanners, both graphic and nongraphic, share. This shared functionality includes methods to add, remove, inspect and test components governed by the spanner, as well as basic formatting properties. The other spanner classes, such as beam and glissando, all inherit from this class and receive the functionality implemented here.

Read-only Properties

`Spanner.components`
 Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Spanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Spanner.leaves

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Spanner.override

LilyPond grob override component plug-in.

Spanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Spanner.prolated_duration

Sum of prolated duration of all components in spanner.

Spanner.set

LilyPond context setting component plug-in.

Spanner.start_offset

Read-only start offset of spanner.

Spanner.stop_offset

Read-only stop offset of spanner.

Spanner.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Spanner.written_duration

Sum of written duration of all components in spanner.

Methods

Spanner.append(*component*)

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Spanner.append_left (*component*)

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Spanner.clear ()

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Spanner.extend (*components*)

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Spanner.extend_left (*components*)

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

`Spanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

`Spanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

`Spanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

`Spanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

`Spanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Special Methods

`Spanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Spanner.**__contains__**(*expr*)

Spanner.**__eq__**(*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Spanner.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Spanner.**__getitem__**(*expr*)

Spanner.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Spanner.**__le__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Spanner.**__len__**()

Spanner.**__lt__**(*other*)

Trivial comparison to allow doctests to work.

Spanner.**__ne__**(*arg*)

True when `id(self)` does not equal `id(arg)`.

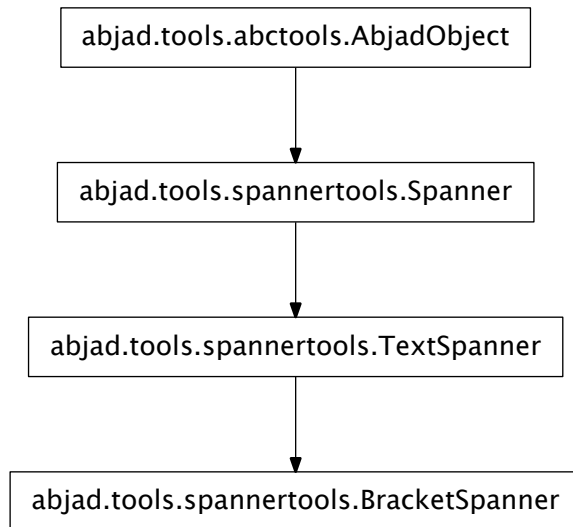
Return boolean.

Inherited from `abctools.AbjadObject`

Spanner.**__repr__**()

concrete classes

spannertools.BracketSpanner



class spannertools.**BracketSpanner** (*components=None*)

Abjad bracket spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> spannertools.BracketSpanner(staff[:])
BracketSpanner(c'8, d'8, e'8, f'8)
```

```
>>> f(staff)
\new Staff {
  \override TextSpanner #'bound-details #'left #'text = \markup {
    \draw-line #'(0 . -1) }
  \override TextSpanner #'bound-details #'left-broken #'text = ##f
  \override TextSpanner #'bound-details #'right #'text = \markup {
    \draw-line #'(0 . -1) }
  \override TextSpanner #'bound-details #'right-broken #'text = ##f
  \override TextSpanner #'color = #red
  \override TextSpanner #'dash-fraction = #1
  \override TextSpanner #'staff-padding = #2
  \override TextSpanner #'thickness = #1.5
  c'8 \startTextSpan
  d'8
  e'8
  f'8 \stopTextSpan
  \revert TextSpanner #'bound-details #'left #'text
  \revert TextSpanner #'bound-details #'left-broken #'text
  \revert TextSpanner #'bound-details #'right #'text
  \revert TextSpanner #'bound-details #'right-broken #'text
```



```

\revert TextSpanner #'color
\revert TextSpanner #'dash-fraction
\revert TextSpanner #'staff-padding
\revert TextSpanner #'thickness
}

```

Render 1.5-unit thick solid red spanner.

Draw nibs at beginning and end of spanner.

Do not draw nibs at line breaks.

Return bracket spanner.

Read-only Properties

BracketSpanner.components

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

BracketSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

BracketSpanner.leaves

Return read-only tuple of leaves in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

BracketSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

BracketSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

BracketSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

BracketSpanner.set

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`BracketSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`BracketSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`BracketSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`BracketSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Methods

`BracketSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`BracketSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`BracketSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
```

```
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`BracketSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`BracketSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`BracketSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
```

```

        d'8 [
        e'8
        f'8 ]
    }

```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`BracketSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}

```

Return list.

Inherited from `spannertools.Spanner`

`BracketSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0

```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`BracketSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`BracketSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`BracketSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`BracketSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`BracketSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`BracketSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`BracketSpanner.__getitem__(expr)`
Inherited from `spannertools.Spanner`

`BracketSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

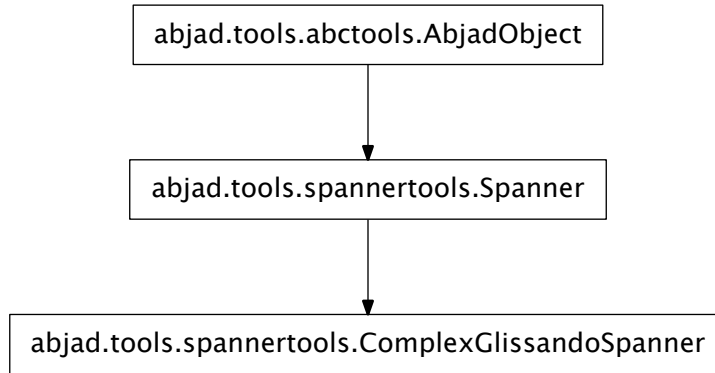
`BracketSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`BracketSpanner.__len__()`
Inherited from `spannertools.Spanner`

`BracketSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.
Inherited from `spannertools.Spanner`

`BracketSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`BracketSpanner.__repr__()`
Inherited from `spannertools.Spanner`

spannertools.ComplexGlissandoSpanner

class spannertools.**ComplexGlissandoSpanner** (*components=None*)
 New in version 2.9. Abjad rest-skipping glissando spanner:

```

>>> staff = Staff("c'16 r r g' r8 c'8")

>>> spannertools.ComplexGlissandoSpanner(staff[:])
ComplexGlissandoSpanner(c'16, r16, r16, g'16, r8, c'8)

>>> f(staff)
\new Staff {
  c'16 \glissando
  \once \override NoteColumn #'glissando-skip = ##t
  \once \override Rest #'transparent = ##t
  r16
  \once \override NoteColumn #'glissando-skip = ##t
  \once \override Rest #'transparent = ##t
  r16
  g'16 \glissando
  \once \override NoteColumn #'glissando-skip = ##t
  \once \override Rest #'transparent = ##t
  r8
  c'8
}
  
```

Should be used with beamtools.BeamSpanner for best effect, along with an override of Stem #'stemlet-length, in order to generate stemlets over each invisible rest.

Format nonlast leaves in spanner with LilyPond glissando command.

Set all Rest instances to transparent.

Set all NoteColumns filled with silences to be skipped by glissandi.

Return *ComplexGlissandoSpanner* instance.

Read-only Properties

`ComplexGlissandoSpanner.components`

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`ComplexGlissandoSpanner.written_duration`
Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Methods

`ComplexGlissandoSpanner.append(component)`
Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return `none`.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.append_left(component)`
Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return `none`.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.clear()`
Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return `none`.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.extend(components)`
Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
```

```
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`ComplexGlissandoSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ComplexGlissandoSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ComplexGlissandoSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`ComplexGlissandoSpanner.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

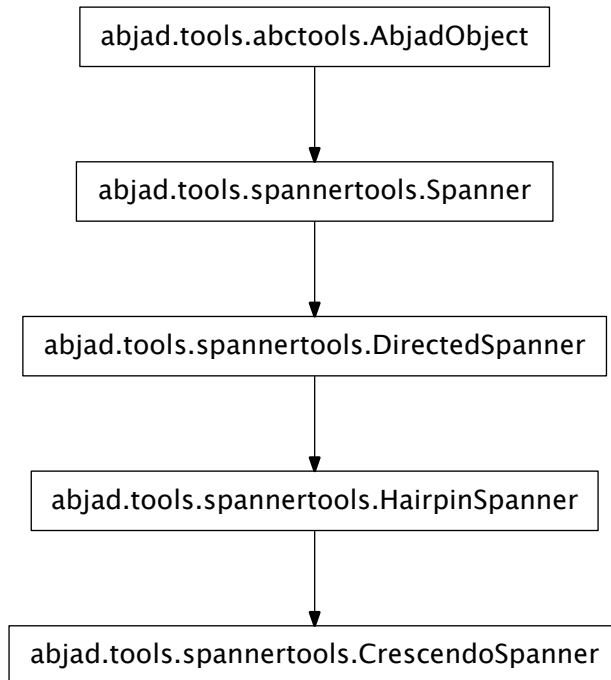
`ComplexGlissandoSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`ComplexGlissandoSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`ComplexGlissandoSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

spannertools.CrescendoSpanner



class spannertools.**CrescendoSpanner** (*components=None, include_rests=True, direction=None*)
 Abjad crescendo spanner that includes rests:

```

>>> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")

>>> f(staff)
\new Staff {
  r4
  c'8
  d'8
  e'8
  f'8
  r4
}

>>> spannertools.CrescendoSpanner(staff[:], include_rests=True)
CrescendoSpanner(r4, c'8, d'8, e'8, f'8, r4)

>>> f(staff)
\new Staff {
  r4 \<
  c'8
  d'8
  e'8
  f'8

```

```

    r4 \!
}

```

Abjad crescendo spanner that does not include rests:

```

>>> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")

>>> f(staff)
\new Staff {
    r4
    c'8
    d'8
    e'8
    f'8
    r4
}

>>> spannertools.CrescendoSpanner(staff[:], include_rests=False)
CrescendoSpanner(r4, c'8, d'8, e'8, f'8, r4)

>>> f(staff)
\new Staff {
    r4
    c'8 \<
    d'8
    e'8
    f'8 \!
    r4
}

```

Return crescendo spanner.

Read-only Properties

`CrescendoSpanner.components`

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`CrescendoSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.leaves`

Return read-only tuple of leaves in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`CrescendoSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`CrescendoSpanner.direction`

Inherited from `spannertools.DirectedSpanner`

`CrescendoSpanner.include_rests`

Get boolean hairpin rests setting:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
>>> hairpin.include_rests
True
```

Set boolean hairpin rests setting:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
>>> hairpin.include_rests = False
>>> hairpin.include_rests
False
```


Set boolean.

Inherited from `spannertools.HairpinSpanner`

`CrescendoSpanner.shape_string`

Get hairpin shape string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.shape_string
'<'
```

Set hairpin shape string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.shape_string = '>'
>>> hairpin.shape_string
'>'
```

Set string.

Inherited from `spannertools.HairpinSpanner`

`CrescendoSpanner.start_dynamic_string`

Get hairpin start dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.start_dynamic_string
'p'
```

Set hairpin start dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.start_dynamic_string = 'mf'
>>> hairpin.start_dynamic_string
'mf'
```

Set string.

Inherited from `spannertools.HairpinSpanner`

`CrescendoSpanner.stop_dynamic_string`

Get hairpin stop dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.stop_dynamic_string
'f'
```

Set hairpin stop dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.stop_dynamic_string = 'mf'
>>> hairpin.stop_dynamic_string
'mf'
```

Set string.

Inherited from `spannertools.HairpinSpanner`

Methods

`CrescendoSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return `none`.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are `Left`, `Right` and `None`.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
```

```

        f'8 ]
    }

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}

```

Return list.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0

```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.pop()`

Remove and return rightmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)

```

Return component.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.pop_left()`

Remove and return leftmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

```

```
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`CrescendoSpanner.__call__(expr)`

New in version 2.9. Call `spanner` on *expr* as a shortcut to extend `spanner`:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying `spanner` and mark attachment syntax.

Return `spanner`.

Inherited from `spannertools.Spanner`

`CrescendoSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`CrescendoSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`CrescendoSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CrescendoSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`CrescendoSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`CrescendoSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

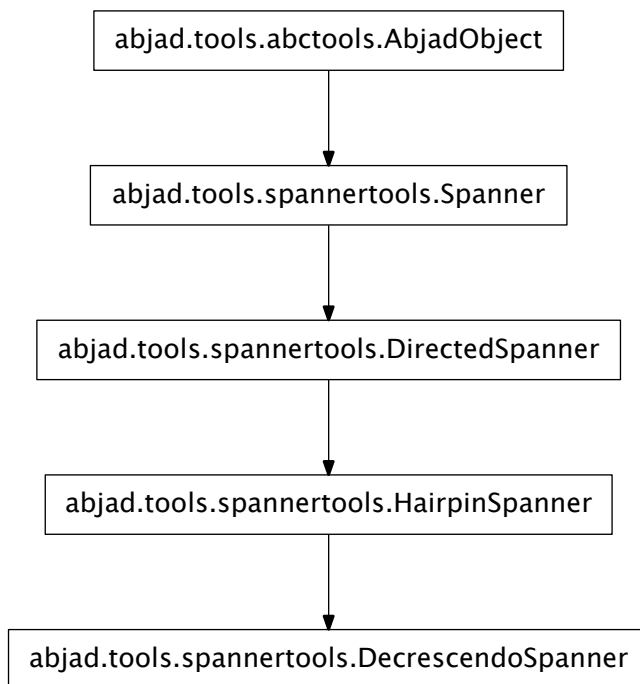
`CrescendoSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`CrescendoSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`CrescendoSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`CrescendoSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

`spannertools.DecrescendoSpanner`



```

class spannertools.DecrescendoSpanner (components=None,      include_rests=True,      direc-
                                     tion=None)
    Abjad decrescendo spanner that includes rests:

    >>> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")

    >>> f(staff)
    \new Staff {
    
```

```

    r4
    c'8
    d'8
    e'8
    f'8
    r4
}

>>> spannertools.DecrescendoSpanner(staff[:], include_rests=True)
DecrescendoSpanner(r4, c'8, d'8, e'8, f'8, r4)

>>> f(staff)
\new Staff {
    r4 \>
    c'8
    d'8
    e'8
    f'8
    r4 \!
}

```

Abjad decrescendo spanner that does not include rests:

```

>>> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")

>>> f(staff)
\new Staff {
    r4
    c'8
    d'8
    e'8
    f'8
    r4
}

>>> spannertools.DecrescendoSpanner(staff[:], include_rests=False)
DecrescendoSpanner(r4, c'8, d'8, e'8, f'8, r4)

>>> f(staff)
\new Staff {
    r4
    c'8 \>
    d'8
    e'8
    f'8 \!
    r4
}

```

Return decrescendo spanner.

Read-only Properties

`DecrescendoSpanner.components`

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

DecrescendoSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

DecrescendoSpanner.leaves

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

DecrescendoSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

DecrescendoSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

DecrescendoSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

DecrescendoSpanner.set

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

DecrescendoSpanner.start_offset

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

DecrescendoSpanner.stop_offset

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

DecrescendoSpanner.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

DecrescendoSpanner.written_duration

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write PropertiesDecrescendoSpanner.**direction**Inherited from `spannertools.DirectedSpanner`DecrescendoSpanner.**include_rests**

Get boolean hairpin rests setting:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
>>> hairpin.include_rests
True
```

Set boolean hairpin rests setting:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
>>> hairpin.include_rests = False
>>> hairpin.include_rests
False
```

Set boolean.

Inherited from `spannertools.HairpinSpanner`DecrescendoSpanner.**shape_string**

Get hairpin shape string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.shape_string
'<'
```

Set hairpin shape string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.shape_string = '>'
>>> hairpin.shape_string
'>'
```

Set string.

Inherited from `spannertools.HairpinSpanner`DecrescendoSpanner.**start_dynamic_string**

Get hairpin start dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.start_dynamic_string
'p'
```

Set hairpin start dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.start_dynamic_string = 'mf'
>>> hairpin.start_dynamic_string
'mf'
```

Set string.

Inherited from `spannertools.HairpinSpanner`

`DecrescendoSpanner.stop_dynamic_string`

Get hairpin stop dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.stop_dynamic_string
'f'
```

Set hairpin stop dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.stop_dynamic_string = 'mf'
>>> hairpin.stop_dynamic_string
'mf'
```

Set string.

Inherited from `spannertools.HairpinSpanner`

Methods

`DecrescendoSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.pop()
Note('f'8)

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.pop_left()`
Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`DecrescendoSpanner.__call__(expr)`
New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__contains__(expr)`
Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DecrescendoSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`DecrescendoSpanner.__getitem__(expr)`
Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

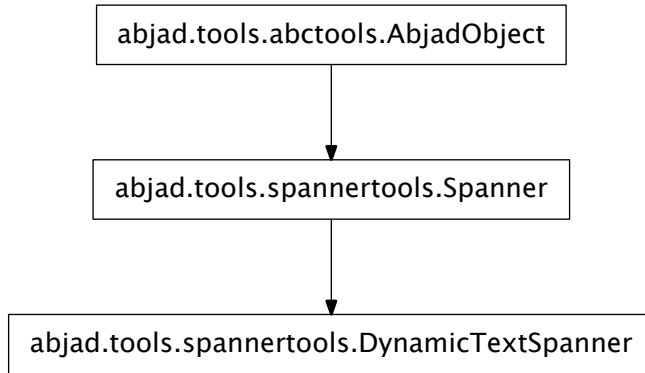
`DecrescendoSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`DecrescendoSpanner.__len__()`
Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.
Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`DecrescendoSpanner.__repr__()`
Inherited from `spannertools.Spanner`

spannertools.DynamicTextSpanner

class spannertools.**DynamicTextSpanner** (*components=None*, *mark=''*)
 Abjad dynamic text spanner:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> spannertools.DynamicTextSpanner(staff[:], 'f')
DynamicTextSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 \f
  d'8
  e'8
  f'8
}
  
```

Format dynamic *mark* at first leaf in spanner.

Return dynamic text spanner.

Read-only Properties

DynamicTextSpanner.components

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
  
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

DynamicTextSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`DynamicTextSpanner.mark`

Get dynamic string:


```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> dynamic_text_spanner = spannertools.DynamicTextSpanner(staff[:], 'f')
>>> dynamic_text_spanner.mark
'f'
```

Set dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> dynamic_text_spanner = spannertools.DynamicTextSpanner(staff[:], 'f')
>>> dynamic_text_spanner.mark = 'p'
>>> dynamic_text_spanner.mark
'p'
```

Set string.

Methods

`DynamicTextSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")
```

```
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.pop_left()
Note("c'8")
```

```
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`DynamicTextSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__len__()`

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

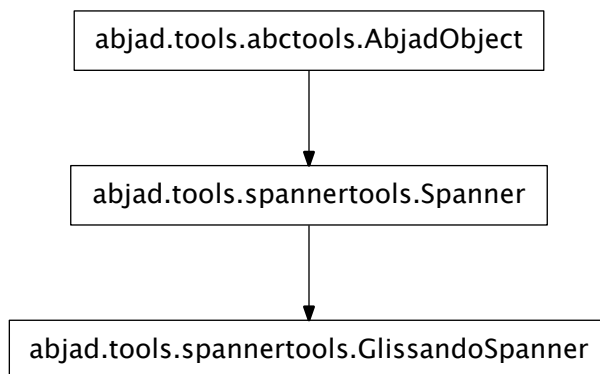
Return boolean.

Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__repr__()`

Inherited from `spannertools.Spanner`

`spannertools.GlissandoSpanner`



class `spannertools.GlissandoSpanner` (*components=None*)

Abjad glissando spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> spannertools.GlissandoSpanner(staff[:])
GlissandoSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 \glissando
  d'8 \glissando
  e'8 \glissando
  f'8
}
```

Format nonlast leaves in spanner with LilyPond `glissando` command.

Return glissando spanner.

Read-only Properties

`GlissandoSpanner.components`

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`GlissandoSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.prolated_duration`
Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.set`
LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.start_offset`
Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.stop_offset`
Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`GlissandoSpanner.written_duration`
Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Methods

`GlissandoSpanner.append(component)`
Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.append_left(component)`
Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```



```
>>> print voice.lilypond_format
\new Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`GlissandoSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`GlissandoSpanner.__contains__(expr)`
Inherited from `spannertools.Spanner`

`GlissandoSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`GlissandoSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`GlissandoSpanner.__getitem__(expr)`
Inherited from `spannertools.Spanner`

`GlissandoSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`GlissandoSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

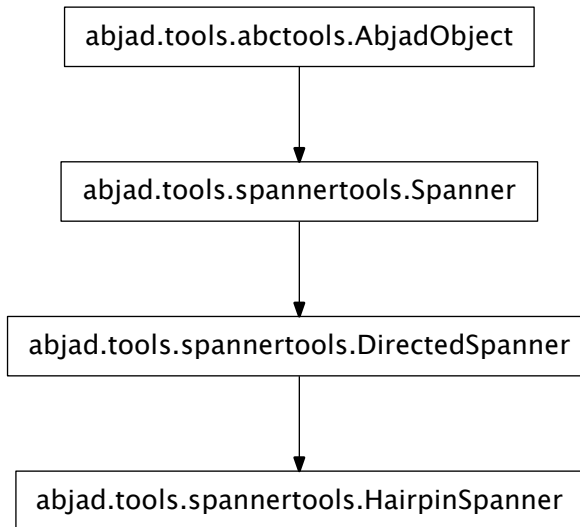
`GlissandoSpanner.__len__()`
Inherited from `spannertools.Spanner`

`GlissandoSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.
Inherited from `spannertools.Spanner`

`GlissandoSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`GlissandoSpanner.__repr__()`
Inherited from `spannertools.Spanner`

spannertools.HairpinSpanner



class spannertools.**HairpinSpanner** (*components=None, descriptor='<', include_rests=True, direction=None*)

Abjad hairpin spanner that includes rests:

```

>>> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")

>>> f(staff)
\new Staff {
    r4
    c'8
    d'8
    e'8
    f'8
    r4
}

>>> spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
HairpinSpanner(r4, c'8, d'8, e'8, f'8, r4)

>>> f(staff)
\new Staff {
    r4 \< \p
    c'8
    d'8
    e'8
    f'8
    r4 \f
}
  
```

Abjad hairpin spanner that does not include rests:

```

>>> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")

>>> f(staff)
\new Staff {
  r4
  c'8
  d'8
  e'8
  f'8
  r4
}

>>> spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = False)
HairpinSpanner(r4, c'8, d'8, e'8, f'8, r4)

>>> f(staff)
\new Staff {
  r4
  c'8 \< \p
  d'8
  e'8
  f'8 \f
  r4
}

```

Return hairpin spanner.

Read-only Properties

HairpinSpanner.components

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

HairpinSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

HairpinSpanner.leaves

Return read-only tuple of leaves in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`HairpinSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`HairpinSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`HairpinSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`HairpinSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`HairpinSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`HairpinSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`HairpinSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`HairpinSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`HairpinSpanner.direction`

Inherited from `spannertools.DirectedSpanner`

`HairpinSpanner.include_rests`

Get boolean hairpin rests setting:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
>>> hairpin.include_rests
True
```

Set boolean hairpin rests setting:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
>>> hairpin.include_rests = False
>>> hairpin.include_rests
False
```

Set boolean.

HairpinSpanner.shape_string

Get hairpin shape string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.shape_string
'<'
```

Set hairpin shape string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.shape_string = '>'
>>> hairpin.shape_string
'>'
```

Set string.

HairpinSpanner.start_dynamic_string

Get hairpin start dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.start_dynamic_string
'p'
```

Set hairpin start dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.start_dynamic_string = 'mf'
>>> hairpin.start_dynamic_string
'mf'
```

Set string.

HairpinSpanner.stop_dynamic_string

Get hairpin stop dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.stop_dynamic_string
'f'
```

Set hairpin stop dynamic string:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
>>> hairpin.stop_dynamic_string = 'mf'
>>> hairpin.stop_dynamic_string
'mf'
```

Set string.

Methods**HairpinSpanner.append** (*component*)Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
```

```
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HairpinSpanner.append_left(component)`
Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HairpinSpanner.clear()`
Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`HairpinSpanner.extend(components)`
Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HairpinSpanner.extend_left(components)`
Add iterable *components* to left of spanner:


```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HairpinSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`HairpinSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`HairpinSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`HairpinSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`HairpinSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`HairpinSpanner.__call__(expr)`

New in version 2.9. Call `spanner` on *expr* as a shortcut to extend `spanner`:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying `spanner` and mark attachment syntax.

Return `spanner`.

Inherited from `spannertools.Spanner`

`HairpinSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`HairpinSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`HairpinSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HairpinSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`HairpinSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`HairpinSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HairpinSpanner.__len__()`

Inherited from `spannertools.Spanner`

`HairpinSpanner.__lt__(other)`

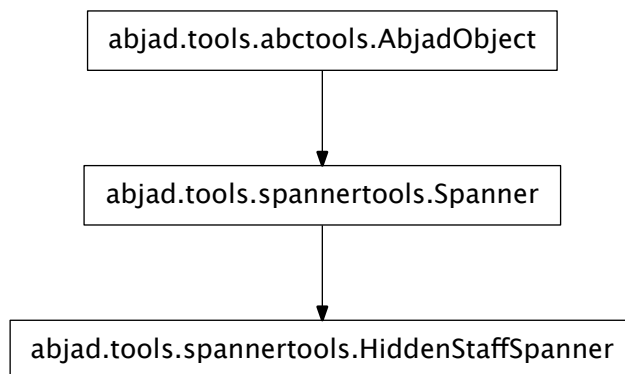
Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

`HairpinSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`HairpinSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

`spannertools.HiddenStaffSpanner`



```

class spannertools.HiddenStaffSpanner (components=None)
    Abjad hidden staff spanner:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> spannertools.HiddenStaffSpanner(staff[:2])
    HiddenStaffSpanner(c'8, d'8)

    >>> f(staff)
    \new Staff {
        \stopStaff
        c'8
        d'8
        \startStaff
        e'8
        f'8
    }
    
```

Hide staff behind leaves in spanner.
 Return hidden staff spanner.

Read-only Properties

`HiddenStaffSpanner.components`
 Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Methods

`HiddenStaffSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
```



```
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`HiddenStaffSpanner.__call__(expr)`

New in version 2.9. Call `spanner` on *expr* as a shortcut to extend `spanner`:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying `spanner` and mark attachment syntax.

Return `spanner`.

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

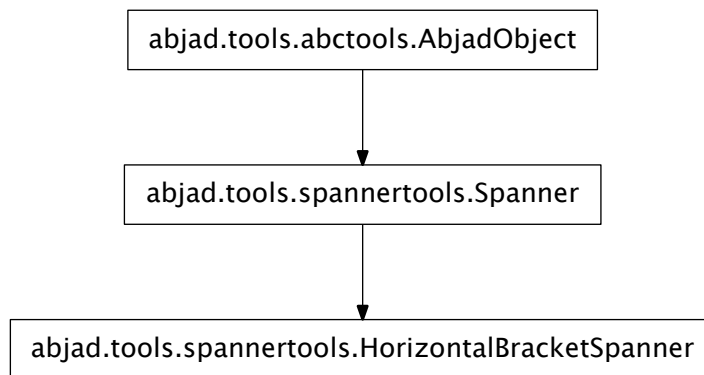
`HiddenStaffSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

`spannertools.HorizontalBracketSpanner`



class `spannertools.HorizontalBracketSpanner` (*components=None*)

New in version 2.4. Abjad horizontal bracket spanner:

```

>>> voice = Voice("c'4 d'4 e'4 f'4")
>>> voice.engraver_consists.append('Horizontal_bracket_engraver')

>>> horizontal_bracket_spanner = spannertools.HorizontalBracketSpanner(voice[:])

>>> horizontal_bracket_spanner
HorizontalBracketSpanner(c'4, d'4, e'4, f'4)
  
```

```
>>> f(voice)
\new Voice \with {
  \consists Horizontal_bracket_engraver
} {
  c'4 \startGroup
  d'4
  e'4
  f'4 \stopGroup
}
```

Return horizontal bracket spanner.

Read-only Properties

HorizontalBracketSpanner.components

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

HorizontalBracketSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

HorizontalBracketSpanner.leaves

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

HorizontalBracketSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

HorizontalBracketSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

HorizontalBracketSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

HorizontalBracketSpanner.set

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Methods

`HorizontalBracketSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.pop()
Note('f'8)

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`HorizontalBracketSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__getitem__(expr)`
Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__len__()`
Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

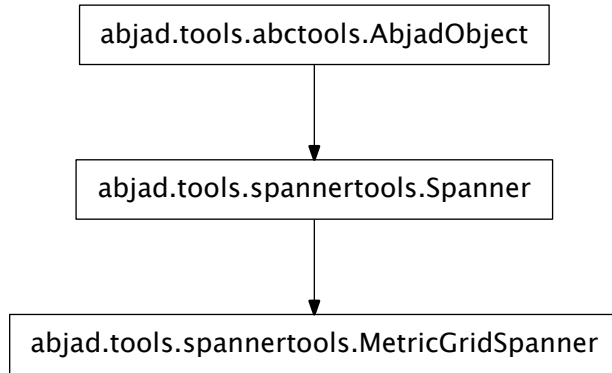
`HorizontalBracketSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__repr__()`
Inherited from `spannertools.Spanner`

spannertools.MetricGridSpanner



class spannertools.**MetricGridSpanner** (*components=None, meters=None*)

Abjad metric grid spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c'8")
```

```
>>> spannertools.MetricGridSpanner(staff.leaves, meters=[(1, 8), (1, 4)])
MetricGridSpanner(c'8, d'8, e'8, f'8, g'8, a'8, b'8, c'8)
```

```
>>> f(staff)
\new Staff {
  \time 1/8
  c'8
  \time 1/4
  d'8
  e'8
  \time 1/8
  f'8
  \time 1/4
  g'8
  a'8
  \time 1/8
  b'8
  \time 1/4
  c'8
}
```

Format leaves in spanner cyclically with *meters*.

Return metric grid spanner.

Read-only Properties

MetricGridSpanner.components

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

MetricGridSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

MetricGridSpanner.leaves

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

MetricGridSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

MetricGridSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

MetricGridSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

MetricGridSpanner.set

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

MetricGridSpanner.start_offset

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

MetricGridSpanner.stop_offset

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

MetricGridSpanner.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`MetricGridSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`MetricGridSpanner.meters`

Get metric grid meters:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c'8")
>>> metric_grid_spanner = spannertools.MetricGridSpanner(
...     staff.leaves, meters=[(1, 8), (1, 4)])
```

Set metric grid meters:

```
>>> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c'8")
>>> metric_grid_spanner = spannertools.MetricGridSpanner(
...     staff.leaves, meters=[(1, 8), (1, 4)])
>>> metric_grid_spanner.meters = [Duration(1, 4)]
```

Set iterable.

Methods

`MetricGridSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.split_on_bar()`

Temporarily unavailable.

`MetricGridSpanner.splitting_condition(leaf)`

User-definable boolean function to determine whether leaf should be split:

```
>>> voice = Voice("c'4 r4 c'4")

>>> f(voice)
\new Voice {
  c'4
  r4
  c'4
}

>>> def cond(leaf):
...     if not isinstance(leaf, Rest): return True
...     else: return False
>>> metric_grid_spanner = spannertools.MetricGridSpanner(
...     voice.leaves, [Duration(1, 8)])
>>> metric_grid_spanner.splitting_condition = cond

>>> metric_grid_spanner.split_on_bar()
```

```
>>> f(voice)
\new Voice {
  \time 1/8
  c'8 ~
  c'8
  r4
  c'8 ~
  c'8
}
```

Function defaults to return true.

Special Methods

`MetricGridSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`MetricGridSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`MetricGridSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`MetricGridSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`MetricGridSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

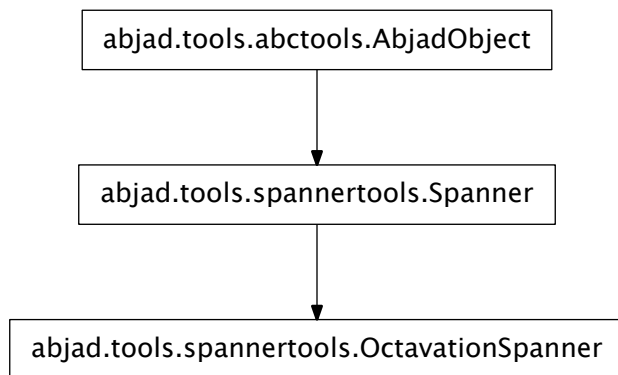
`MetricGridSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

`spannertools.OctavationSpanner`



```

class spannertools.OctavationSpanner (components=None, start=0, stop=0)
  Abjad octavation spanner:

  >>> staff = Staff("c'8 d'8 e'8 f'8")

  >>> spanner = spannertools.OctavationSpanner(staff[:, start=1)

  >>> f(staff)
  \new Staff {
    \ottava #1
    c'8
    d'8
  
```



```
e'8
f'8
\ottava #0
}
```

Return octavation spanner.

Read-only Properties

`OctavationSpanner.components`

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`OctavationSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`OctavationSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`OctavationSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`OctavationSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`OctavationSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`OctavationSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`OctavationSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`OctavationSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`OctavationSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`OctavationSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`OctavationSpanner.start`

Get octavation start:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> octavation = spannertools.OctavationSpanner(staff[:], start=1)
>>> octavation.start
1
```

Set octavation start:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> octavation = spannertools.OctavationSpanner(staff[:], start=1)
>>> octavation.start
1
```

Set integer.

`OctavationSpanner.stop`

Get octavation stop:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> octavation = spannertools.OctavationSpanner(staff[:], start=2, stop=1)
>>> octavation.stop
1
```

Set octavation stop:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> octavation = spannertools.OctavationSpanner(staff[:], start=2, stop=1)
>>> octavation.stop = 0
>>> octavation.stop
0
```

Set integer.

Methods

`OctavationSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`OctavationSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`OctavationSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`OctavationSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`OctavationSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`OctavationSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`OctavationSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
  c'8 [
```

```

        d'8
        e'8
        f'8 ]
    }

```

Return list.

Inherited from `spannertools.Spanner`

`OctavationSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0

```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`OctavationSpanner.pop()`

Remove and return rightmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)

```

Return component.

Inherited from `spannertools.Spanner`

`OctavationSpanner.pop_left()`

Remove and return leftmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)

```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`OctavationSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`OctavationSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`OctavationSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`OctavationSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OctavationSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`OctavationSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OctavationSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OctavationSpanner.__len__()`

Inherited from `spannertools.Spanner`

`OctavationSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

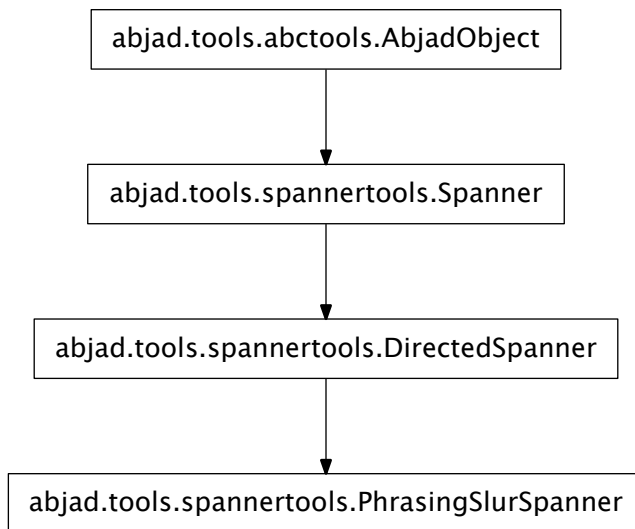
```
OctavationSpanner.__ne__(arg)
    True when id(self) does not equal id(arg).

    Return boolean.

    Inherited from abctools.AbjadObject

OctavationSpanner.__repr__()
    Inherited from spannertools.Spanner
```

spannertools.PhrasingSlurSpanner



```
class spannertools.PhrasingSlurSpanner(components=None, direction=None)
    Abjad phrasing slur spanner:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> spannertools.PhrasingSlurSpanner(staff[:])
    PhrasingSlurSpanner(c'8, d'8, e'8, f'8)

    >>> f(staff)
    \new Staff {
      c'8 \{
        d'8
        e'8
        f'8 \}
    }
```

Return phrasing slur spanner.

Read-only Properties

PhrasingSlurSpanner.components

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.leaves

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.set

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.start_offset

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.stop_offset

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`PhrasingSlurSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`PhrasingSlurSpanner.direction`

Inherited from `spannertools.DirectedSpanner`

Methods

`PhrasingSlurSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return `None`.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return `None`.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return `None`.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return `none`.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return `none`.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are `Left`, `Right` and `None`.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`PhrasingSlurSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`PhrasingSlurSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PhrasingSlurSpanner.__getitem__(expr)`
 Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`PhrasingSlurSpanner.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

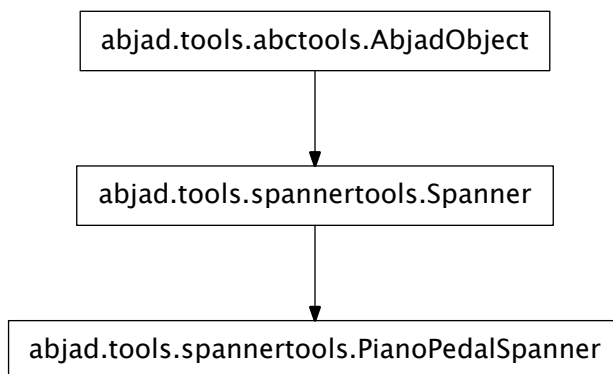
`PhrasingSlurSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`PhrasingSlurSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

`spannertools.PianoPedalSpanner`



```

class spannertools.PianoPedalSpanner(components=None)
    Abjad piano pedal spanner:
  
```

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> spannertools.PianoPedalSpanner(staff[:])
PianoPedalSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  \set Staff.pedalSustainStyle = #'mixed
  c'8 \sustainOn
  d'8
  e'8
  f'8 \sustainOff
}
```

Return piano pedal spanner.

Read-only Properties

PianoPedalSpanner.components

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

PianoPedalSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

PianoPedalSpanner.leaves

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use `spanner-` specific iteration tools.

Inherited from `spannertools.Spanner`

PianoPedalSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

PianoPedalSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

PianoPedalSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`PianoPedalSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`PianoPedalSpanner.kind`

Get piano pedal spanner kind:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> spanner = spannertools.PianoPedalSpanner(staff[:])
>>> spanner.kind
'sustain'
```

Set piano pedal spanner kind:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> spanner = spannertools.PianoPedalSpanner(staff[:])
>>> spanner.kind = 'sostenuto'
>>> spanner.kind
'sostenuto'
```

Acceptable values 'sustain', 'sostenuto', 'corda'.

`PianoPedalSpanner.style`

Get piano pedal spanner style:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> spanner = spannertools.PianoPedalSpanner(staff[:])
>>> spanner.style
'mixed'
```

Set piano pedal spanner style:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> spanner = spannertools.PianoPedalSpanner(staff[:])
>>> spanner.style = 'bracket'
>>> spanner.style
'bracket'
```

Acceptable values 'mixed', 'bracket', 'text'.

Methods

PianoPedalSpanner.**append**(*component*)

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

PianoPedalSpanner.**append_left**(*component*)

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

PianoPedalSpanner.**clear**()

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

PianoPedalSpanner.**extend**(*components*)

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return `none`.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are `Left`, `Right` and `None`.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
```

```

        f'8 ]
    }

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}

```

Return list.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0

```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.pop()`

Remove and return rightmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)

```

Return component.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.pop_left()`

Remove and return leftmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

```

```
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`PianoPedalSpanner.__call__(expr)`

New in version 2.9. Call `spanner` on *expr* as a shortcut to extend `spanner`:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying `spanner` and mark attachment syntax.

Return `spanner`.

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`PianoPedalSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`PianoPedalSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`PianoPedalSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`PianoPedalSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

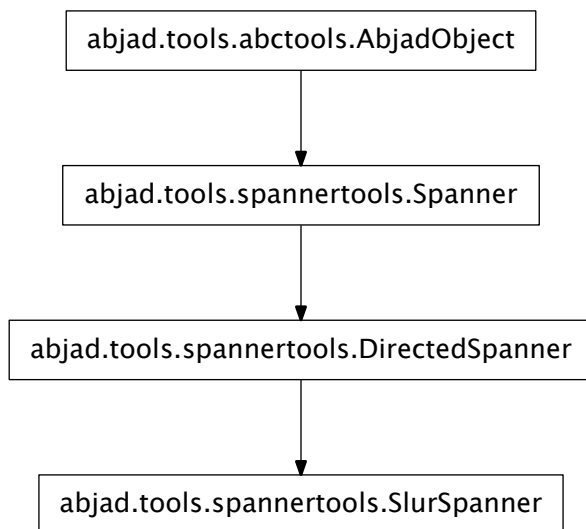
`PianoPedalSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`PianoPedalSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`PianoPedalSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`PianoPedalSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

`spannertools.SlurSpanner`



```

class spannertools.SlurSpanner(components=None, direction=None)
    Abjad slur spanner:

    >>> staff = Staff("c'8 d'8 e'8 f'8")

    >>> spannertools.SlurSpanner(staff[:])
    SlurSpanner(c'8, d'8, e'8, f'8)

    >>> f(staff)
    \new Staff {
        c'8 (
        d'8
        e'8
    
```

```
f'8 )
}
```

Return slur spanner.

Read-only Properties

`SlurSpanner.components`

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`SlurSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`SlurSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

`SlurSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`SlurSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`SlurSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`SlurSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`SlurSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`SlurSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`SlurSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`SlurSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`SlurSpanner.direction`

Inherited from `spannertools.DirectedSpanner`

Methods

`SlurSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`SlurSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`SlurSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`SlurSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`SlurSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`SlurSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`SlurSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`SlurSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`SlurSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```



```
>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

```
SlurSpanner.pop_left()
Remove and return leftmost component in spanner:

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`SlurSpanner.__call__(expr)`
 New in version 2.9. Call `spanner` on *expr* as a shortcut to extend `spanner`:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying `spanner` and mark attachment syntax.

Return `spanner`.

Inherited from `spannertools.Spanner`

`SlurSpanner.__contains__(expr)`
 Inherited from `spannertools.Spanner`

`SlurSpanner.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SlurSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SlurSpanner.__getitem__(expr)`
Inherited from `spannertools.Spanner`

`SlurSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SlurSpanner.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SlurSpanner.__len__()`
Inherited from `spannertools.Spanner`

`SlurSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

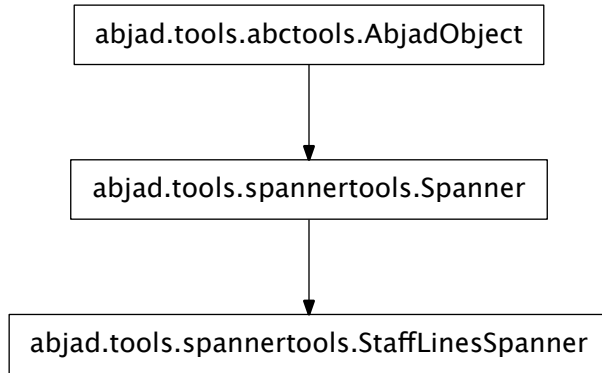
`SlurSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SlurSpanner.__repr__()`
Inherited from `spannertools.Spanner`

spannertools.StaffLinesSpanner



class spannertools.**StaffLinesSpanner** (*components=None, lines=5*)

Abjad staff lines spanner:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> spannertools.StaffLinesSpanner(staff[:2], 1)
StaffLinesSpanner(c'8, d'8)

>>> f(staff)
\new Staff {
  \stopStaff
  \override Staff.StaffSymbol #'line-count = #1
  \startStaff
  c'8
  d'8
  \stopStaff
  \revert Staff.StaffSymbol #'line-count
  \startStaff
  e'8
  f'8
}
  
```

Staff lines spanner handles changing either the line-count or the line-positions property of the StaffSymbol grob, as well as automatically stopping and restarting the staff so that the change may take place.

Return staff lines spanner.

Read-only Properties

StaffLinesSpanner.**components**

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
  
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

StaffLinesSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

StaffLinesSpanner.leaves

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

StaffLinesSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

StaffLinesSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

StaffLinesSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

StaffLinesSpanner.set

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

StaffLinesSpanner.start_offset

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

StaffLinesSpanner.stop_offset

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

StaffLinesSpanner.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

StaffLinesSpanner.written_duration

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties**StaffLinesSpanner.lines**

Get staff lines spanner line count:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> spanner = spannertools.StaffLinesSpanner(staff[:2], 1)
>>> spanner.lines
1
```

Set staff lines spanner line count:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> spanner = spannertools.StaffLinesSpanner(staff[:2], 1)
>>> spanner.lines = 2
>>> spanner.lines
2
```

Set integer.

Methods**StaffLinesSpanner.append**(*component*)Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`**StaffLinesSpanner.append_left**(*component*)Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`**StaffLinesSpanner.clear**()

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`StaffLinesSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`StaffLinesSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`StaffLinesSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`StaffLinesSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`StaffLinesSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`StaffLinesSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.pop()
Note('f'8)

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`StaffLinesSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`StaffLinesSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`StaffLinesSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`StaffLinesSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`StaffLinesSpanner.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`StaffLinesSpanner.__getitem__(expr)`
 Inherited from `spannertools.Spanner`

`StaffLinesSpanner.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`StaffLinesSpanner.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

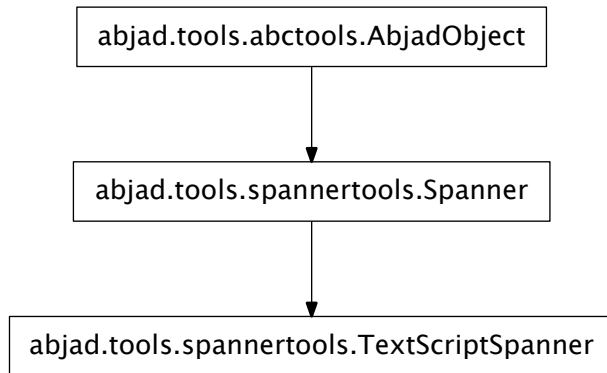
`StaffLinesSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`StaffLinesSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`StaffLinesSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`StaffLinesSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

spannertools.TextScriptSpanner



class spannertools.**TextScriptSpanner** (*components=None*)

New in version 2.0. Abjad text script spanner:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> spanner = spannertools.TextScriptSpanner(staff[:])
>>> spanner.override.text_script.color = 'red'
>>> markuptools.Markup(r'\italic { espressivo }', Up)(staff[1])
Markup((MarkupCommand('italic', ['espressivo']),), direction=Up)(d'8)

>>> f(staff)
\new Staff {
  \override TextScript #'color = #red
  c'8
  d'8 ^ \markup { \italic { espressivo } }
  e'8
  f'8
  \revert TextScript #'color
}
  
```

Override LilyPond TextScript grob.

Return text script spanner.

Read-only Properties

TextScriptSpanner.components

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
  
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

TextScriptSpanner.duration_in_seconds
Sum of duration of all leaves in spanner, in seconds.
Inherited from `spannertools.Spanner`

TextScriptSpanner.leaves
Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use `spanner-` specific iteration tools.

Inherited from `spannertools.Spanner`

TextScriptSpanner.override
LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

TextScriptSpanner.preprolated_duration
Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

TextScriptSpanner.prolated_duration
Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

TextScriptSpanner.set
LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

TextScriptSpanner.start_offset
Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

TextScriptSpanner.stop_offset
Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

TextScriptSpanner.storage_format
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

TextScriptSpanner.written_duration
Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Methods

`TextScriptSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return `none`.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are `Left`, `Right` and `None`.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
```

```

        f'8 ]
    }

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}

```

Return list.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0

```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.pop()`

Remove and return rightmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)

```

Return component.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.pop_left()`

Remove and return leftmost component in spanner:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

```

```
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`TextScriptSpanner.__call__(expr)`

New in version 2.9. Call `spanner` on *expr* as a shortcut to extend `spanner`:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying `spanner` and mark attachment syntax.

Return `spanner`.

Inherited from `spannertools.Spanner`

`TextScriptSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`TextScriptSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TextScriptSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TextScriptSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`TextScriptSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TextScriptSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

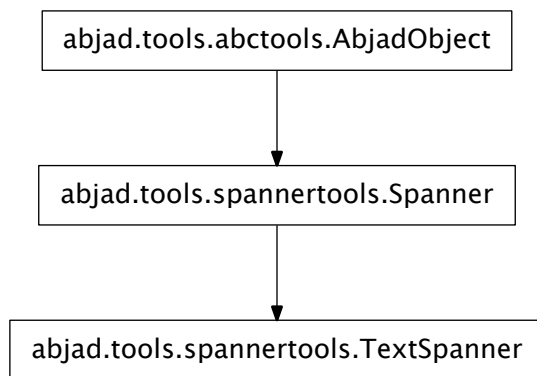
`TextScriptSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`TextScriptSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`TextScriptSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`TextScriptSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

`spannertools.TextSpanner`



```

class spannertools.TextSpanner (components=None)
    New in version 2.0. Abjad text spanner:

    >>> staff = Staff("c'8 d'8 e'8 f'8")
    >>> text_spanner = spannertools.TextSpanner(staff[:])

    >>> markup = markuptools.Markup(markuptools.MarkupCommand(
    ...     'bold', markuptools.MarkupCommand('italic', 'foo'))
    >>> text_spanner.override.text_spanner.bound_details__left__text = markup
    >>> markup = markuptools.Markup(
    ...     markuptools.MarkupCommand('draw-line', schemetools.SchemePair(0, -1)))
    >>> text_spanner.override.text_spanner.bound_details__right__text = markup
    >>> text_spanner.override.text_spanner.dash_fraction = 1

    >>> f(staff)
    \new Staff {
        \override TextSpanner #'bound-details #'left #'text = \markup {
            \bold \italic foo }
    
```



```

\override TextSpanner #'bound-details #'right #'text = \markup {
  \draw-line #'(0 . -1) }
\override TextSpanner #'dash-fraction = #1
c'8 \startTextSpan
d'8
e'8
f'8 \stopTextSpan
\revert TextSpanner #'bound-details #'left #'text
\revert TextSpanner #'bound-details #'right #'text
\revert TextSpanner #'dash-fraction
}

```

Override LilyPond TextSpanner grob.

Return text spanner.

Read-only Properties

TextSpanner.components

Return read-only tuple of components in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

TextSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

TextSpanner.leaves

Return read-only tuple of leaves in spanner.

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Inherited from `spannertools.Spanner`

TextSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

TextSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

TextSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`TextSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`TextSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`TextSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`TextSpanner.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`TextSpanner.written_duration`

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Methods

`TextSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TextSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TextSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`TextSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TextSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TextSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`TextSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`TextSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`TextSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`TextSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`TextSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`TextSpanner.__contains__(expr)`
Inherited from `spannertools.Spanner`

`TextSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TextSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TextSpanner.__getitem__(expr)`
Inherited from `spannertools.Spanner`

`TextSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TextSpanner.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TextSpanner.__len__()`
Inherited from `spannertools.Spanner`

`TextSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

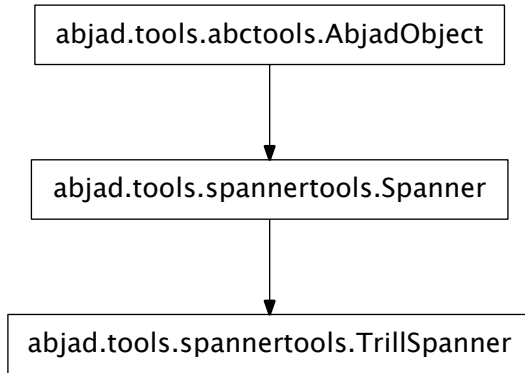
`TextSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TextSpanner.__repr__()`
Inherited from `spannertools.Spanner`

spannertools.TrillSpanner



class spannertools.**TrillSpanner** (*components=None*)
 Abjad trill spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
```

```
>>> spannertools.TrillSpanner(staff[:])
TrillSpanner(c'8, d'8, e'8, f'8)
```

```
>>> f(staff)
\new Staff {
  c'8 \startTrillSpan
  d'8
  e'8
  f'8 \stopTrillSpan
}
```

Override LilyPond TrillSpanner grob.

Return trill spanner.

Read-only Properties

TrillSpanner.components

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from spannertools.Spanner

TrillSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Inherited from spannertools.Spanner

TrillSpanner.leaves

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use `spanner-` specific iteration tools.

Inherited from `spannertools.Spanner`

TrillSpanner.override

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

TrillSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

TrillSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

TrillSpanner.set

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

TrillSpanner.start_offset

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

TrillSpanner.stop_offset

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

TrillSpanner.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

TrillSpanner.written_duration

Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

TrillSpanner.pitch

Optional read / write pitch for pitched trills.

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> trill = spannertools.TrillSpanner(t[:2])
>>> trill.pitch = pitchtools.NamedChromaticPitch('cs', 4)
```



```
>>> f(t)
\new Staff {
  \pitchedTrill c'8 \startTrillSpan cs'
  d'8 \stopTrillSpan
  e'8
  f'8
}
```

Set pitch.

`TrillSpanner.written_pitch`

Methods

`TrillSpanner.append(component)`

Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TrillSpanner.append_left(component)`

Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TrillSpanner.clear()`

Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`TrillSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TrillSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TrillSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`TrillSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`TrillSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`TrillSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")

>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`TrillSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop_left()
Note("c'8")

>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`TrillSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`TrillSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`TrillSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TrillSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TrillSpanner.__getitem__(expr)`
 Inherited from `spannertools.Spanner`

`TrillSpanner.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`TrillSpanner.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`TrillSpanner.__len__()`
 Inherited from `spannertools.Spanner`

`TrillSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.
 Inherited from `spannertools.Spanner`

`TrillSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`TrillSpanner.__repr__()`
 Inherited from `spannertools.Spanner`

functions

`spannertools.all_are_spanners`

`spannertools.all_are_spanners(expr)`
 New in version 2.6. True when *expr* is a sequence of Abjad spanners:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])

>>> spannertools.all_are_spanners([spanner])
True
```

True when *expr* is an empty sequence:

```
>>> spannertools.all_are_spanners([])
True
```

Otherwise false:

```
>>> spannertools.all_are_spanners('foo')
False
```

Return boolean.

`spannertools.apply_octavation_spanner_to_pitched_components`

`spannertools.apply_octavation_spanner_to_pitched_components` (*expr*, *ottava_numbered_diatonic_pitch=None*, *quinde-*
cisima_numbered_diatonic_pitch=None)

New in version 1.1. Apply octavation spanner to pitched components in *expr*:

```
>>> t = Measure((4, 8), notetools.make_notes([24, 26, 27, 29], [(1, 8)]))
>>> spannertools.apply_octavation_spanner_to_pitched_components(
...     t, ottava_numbered_diatonic_pitch=14)
OctavationSpanner(|4/8(4)|)

>>> f(t)
{
    \time 4/8
    \ottava #1
    c'''8
    d'''8
    ef'''8
    f'''8
    \ottava #0
}
```

Apply octavation spanner according to the diatonic pitch number of the maximum pitch in *expr*.

Return octavation spanner.

`spannertools.destroy_spanners_attached_to_component`

`spannertools.destroy_spanners_attached_to_component` (*component*, *klass=None*)

New in version 1.1. Destroy spanners of *klass* attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> trill = spannertools.TrillSpanner(staff)

>>> f(staff)
\new Staff {
    c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

>>> spanners = spannertools.destroy_spanners_attached_to_component(staff[0])

>>> f(staff)
\new Staff {
    c'8 \startTrillSpan
    d'8
    e'8
    f'8 \stopTrillSpan
}
```

Destroy all spanners when *klass* is none.

Return tuple of zero or more empty spanners.

Order of spanners in return value can not be predicted.

spannertools.destroy_spanners_attached_to_components_in_expr

`spannertools.destroy_spanners_attached_to_components_in_expr(expr, klass=None)`

New in version 2.9. Destroy spanners of *klass* attached to components in *expr*:

```
>>> staff = Staff("c'4 [ ( d' e' f' ) ]")

>>> f(staff)
\new Staff {
  c'4 [ (
    d'4
    e'4
    f'4 ] )
}
```

```
>>> spanners = spannertools.destroy_spanners_attached_to_components_in_expr(staff)

>>> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
}
```

Return tuple of zero or more empty spanners.

Order of spanners in return value can not be predicted.

spannertools.find_index_of_spanner_component_at_score_offset

`spannertools.find_index_of_spanner_component_at_score_offset(spanner, score_offset)`

Return index of component in ‘spanner’ that begins at exactly ‘score_offset’:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)

>>> f(staff)
\new Staff {
  c'8 [
    d'8
    e'8
    f'8 ]
}
```

```
>>> spannertools.find_index_of_spanner_component_at_score_offset(beam, Duration(3, 8))
3
```

Raise spanner population error when no component in *spanner* begins at exactly *score_offset*. Changed in version 2.0: renamed `spannertools.find_index_at_score_offset()` to `spannertools.find_index_of_spanner_component_at_score_offset()`.

`spannertools.find_spanner_component_starting_at_exactly_score_offset`

`spannertools.find_spanner_component_starting_at_exactly_score_offset` (*spanner*, *score_offset*)

Find *spanner* component starting at exactly *score_offset*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)

>>> f(staff)
\new Staff {
  c'8 [
    d'8
    e'8
    f'8 ]
}

>>> spannertools.find_spanner_component_starting_at_exactly_score_offset (
...     beam, Duration(3, 8))
Note("f'8")
```

When no *spanner* component starts at exactly *score_offset* return none.

Return *spanner* component or none. Changed in version 2.0: re-named `spannertools.find_component_at_score_offset()` to `spannertools.find_spanner_component_starting_at_exactly_score_offset()`.

`spannertools.fracture_spanners_attached_to_component`

`spannertools.fracture_spanners_attached_to_component` (*component*, *direction=None*, *klass=None*)

New in version 1.1. Fracture all spanners attached to *component* according to *direction*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> trill = spannertools.TrillSpanner(staff)
>>> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

>>> parts = spannertools.fracture_spanners_attached_to_component(staff[1], Right)

>>> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
    d'8 ] )
  e'8 [ (
    f'8 ] ) \stopTrillSpan
}
```

Set *direction* to Left, Right or None.

spannertools.fracture_spanners_that_cross_components

`spannertools.fracture_spanners_that_cross_components` (*components*)

Fracture to the left of the leftmost component. Fracture to the right of the rightmost component. Do not fracture spanners of any components at higher levels of score. Do not fracture spanners of any components at lower levels of score. Return components.

Components must be thread-contiguous. Some spanners may copy during fracture. This helper is public-safe.

Example:

```
>>> t = Staff(Container(notetools.make_repeated_notes(2)) * 3)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(t)
>>> crescendo = spannertools.CrescendoSpanner(t)
>>> beam = beamtools.BeamSpanner(t[:])
>>> trill = spannertools.TrillSpanner(t.leaves)

>>> f(t)
\new Staff {
  {
    c'8 [ \< \startTrillSpan
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8 ] \! \stopTrillSpan
  }
}

>>> spannertools.fracture_spanners_that_cross_components(t[1:2])
[{'e'8, 'f'8}]

>>> f(t)
\new Staff {
  {
    c'8 [ \< \startTrillSpan
    d'8 ]
  }
  {
    e'8 [
    f'8 ]
  }
  {
    g'8 [
    a'8 ] \! \stopTrillSpan
  }
}
```

Changed in version 2.0: renamed `spannertools.fracture_crossing()` to `spannertools.fracture_spanners_that_cross_components()`.

spannertools.get_nth_leaf_in_spanner

`spannertools.get_nth_leaf_in_spanner` (*spanner, idx*)

Get nth leaf in spanner, no matter how complicated the nesting situation. Changed in version 2.0: renamed

`spannertools.get_nth_leaf()` to `spannertools.get_nth_leaf_in_spanner()`.

`spannertools.get_spanners_attached_to_any_improper_child_of_component`

`spannertools.get_spanners_attached_to_any_improper_child_of_component` (*component*, *klass=None*)

New in version 2.0. Get all spanners attached to any improper children of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> first_slur = spannertools.SlurSpanner(staff.leaves[:2])
>>> second_slur = spannertools.SlurSpanner(staff.leaves[2:])
>>> trill = spannertools.TrillSpanner(staff)

>>> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
  d'8 )
  e'8 (
  f'8 ] ) \stopTrillSpan
}

>>> len(spannertools.get_spanners_attached_to_any_improper_child_of_component(staff))
4
```

Get all spanners of *klass* attached to any proper children of *component*:

```
>>> spanner_klass = spannertools.SlurSpanner
>>> result = spannertools.get_spanners_attached_to_any_proper_child_of_component(
... staff, spanner_klass)

>>> list(sorted(result))
[SlurSpanner(c'8, d'8), SlurSpanner(e'8, f'8)]
```

Get all spanners of any *klass* attached to any proper children of *component*:

```
>>> spanner_klasses = (spannertools.SlurSpanner, beamtools.BeamSpanner)
>>> result = spannertools.get_spanners_attached_to_any_proper_child_of_component(
... staff, spanner_klasses)

>>> list(sorted(result))
[BeamSpanner(c'8, d'8, e'8, f'8), SlurSpanner(c'8, d'8), SlurSpanner(e'8, f'8)]
```

Return unordered set of zero or more spanners.

`spannertools.get_spanners_attached_to_any_improper_parent_of_component`

`spannertools.get_spanners_attached_to_any_improper_parent_of_component` (*component*, *klass=None*)

New in version 1.1. Get all spanners attached to improper parentage of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> trill = spannertools.TrillSpanner(staff)
>>> f(staff)
\new Staff {
```

```

    c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

>>> result = list(sorted(
... spannertools.get_spanners_attached_to_any_improper_parent_of_component(staff[0]))

>>> for spanner in result:
...     spanner
...
BeamSpanner(c'8, d'8, e'8, f'8)
SlurSpanner(c'8, d'8, e'8, f'8)
TrillSpanner({c'8, d'8, e'8, f'8})

```

Return unordered set of zero or more spanners. Changed in version 2.0: renamed `spannertools.get_all_spanners_attached_to_improper_parentage_of_component()` to `spannertools.get_spanners_attached_to_any_improper_parent_of_component()`.

`spannertools.get_spanners_attached_to_any_proper_child_of_component`

`spannertools.get_spanners_attached_to_any_proper_child_of_component` (*component*, *klass=None*)

New in version 2.0. Get all spanners attached to any proper children of *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> first_slur = spannertools.SlurSpanner(staff.leaves[:2])
>>> second_slur = spannertools.SlurSpanner(staff.leaves[2:])
>>> trill = spannertools.TrillSpanner(staff)

>>> f(staff)
\new Staff {
    c'8 [ ( \startTrillSpan
    d'8 )
    e'8 (
    f'8 ] ) \stopTrillSpan
}

>>> len(spannertools.get_spanners_attached_to_any_proper_child_of_component(staff))
3

```

Get all spanners of *klass* attached to any proper children of *component*:

```

>>> spanner_klass = spannertools.SlurSpanner
>>> result = spannertools.get_spanners_attached_to_any_proper_child_of_component(
... staff, spanner_klass)

>>> list(sorted(result))
[SlurSpanner(c'8, d'8), SlurSpanner(e'8, f'8)]

```

Get all spanners of any *klass* attached to any proper children of *component*:

```

>>> spanner_klasses = (spannertools.SlurSpanner, beamtools.BeamSpanner)
>>> result = spannertools.get_spanners_attached_to_any_proper_child_of_component(
... staff, spanner_klasses)

```

```
>>> list(sorted(result))
[BeamSpanner(c'8, d'8, e'8, f'8), SlurSpanner(c'8, d'8), SlurSpanner(e'8, f'8)]
```

Return unordered set of zero or more spanners. Changed in version 2.0: renamed `spannertools.get_all_spanners_attached_to_any_proper_children_of_component()` to `spannertools.get_spanners_attached_to_any_proper_child_of_component()`.

spannertools.get_spanners_attached_to_any_proper_parent_of_component

`spannertools.get_spanners_attached_to_any_proper_parent_of_component` (*component*, *klass=None*)

New in version 2.0. Get all spanners attached to any proper parent of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> trill = spannertools.TrillSpanner(staff)
>>> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
  d'8
  e'8
  f'8 ] ) \stopTrillSpan
}

>>> spannertools.get_spanners_attached_to_any_proper_parent_of_component(staff[0])
set([TrillSpanner({c'8, d'8, e'8, f'8})])
```

Return unordered set of zero or more spanners. Changed in version 2.0: renamed `spannertools.get_all_spanners_attached_to_any_proper_parent_of_component()` to `spannertools.get_spanners_attached_to_any_proper_parent_of_component()`.

spannertools.get_spanners_attached_to_component

`spannertools.get_spanners_attached_to_component` (*component*, *klass=None*)

New in version 2.0. Get all spanners attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> first_slur = spannertools.SlurSpanner(staff.leaves[:2])
>>> second_slur = spannertools.SlurSpanner(staff.leaves[2:])
>>> crescendo = spannertools.CrescendoSpanner(staff.leaves)

>>> f(staff)
\new Staff {
  c'8 [ \< (
  d'8 )
  e'8 (
  f'8 ] \! )
}

>>> result = spannertools.get_spanners_attached_to_component(staff.leaves[0])
>>> for x in sorted(result):
...     x
...
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
CrescendoSpanner(c'8, d'8, e'8, f'8)
SlurSpanner(c'8, d'8)
```

Get spanners of *klass* attached to *component*:

```
>>> klass = beamtools.BeamSpanner
>>> result = spannertools.get_spanners_attached_to_component(staff.leaves[0], klass)
>>> for x in sorted(result):
...     x
...
BeamSpanner(c'8, d'8, e'8, f'8)
```

Get spanners of any *klass* attached to *component*:

```
>>> classes = (beamtools.BeamSpanner, spannertools.SlurSpanner)
>>> result = spannertools.get_spanners_attached_to_component(staff.leaves[0], classes)
>>> for x in sorted(result):
...     x
...
BeamSpanner(c'8, d'8, e'8, f'8)
SlurSpanner(c'8, d'8)
```

Return unordered set of zero or more spanners. Changed in version 2.0: re-named `spannertools.get_all_spanners_attached_to_component()` to `spannertools.get_spanners_attached_to_component()`.

`spannertools.get_spanners_contained_by_components`

`spannertools.get_spanners_contained_by_components` (*components*)

Return unordered set of spanners contained within any component in list of thread-contiguous components. Getter for `t.spanners.contained` across thread-contiguous components. Changed in version 2.0: renamed `spannertools.get_contained()` to `spannertools.get_spanners_contained_by_components()`.

`spannertools.get_spanners_covered_by_components`

`spannertools.get_spanners_covered_by_components` (*components*)

New in version 1.1. Get spanners covered by *components*.

Return unordered set of spanners completely contained within the time bounds of thread-contiguous components.

A spanner *p* is covered by timespan *t* when and only when `t.start_offset <= p.start_offset` and `p.stop_offset <= t.stop_offset`.

Changed in version 2.0: renamed `spannertools.get_covered()` to `spannertools.get_spanners_covered_by_components()`.

`spannertools.get_spanners_on_components_or_component_children`

`spannertools.get_spanners_on_components_or_component_children` (*components*)

Return unordered set of all spanners attaching to any component in *components* or attaching to any of the children of any of the components in *components*. Changed in version 2.0: renamed `spannertools.get_attached()` to `spannertools.get_spanners_on_components_or_component_children()`.

spannertools.get_spanners_that_cross_components

spannertools.get_spanners_that_cross_components (*components*)

Assert thread-contiguous components. Collect spanners that attach to any component in ‘components’. Return unordered set of crossing spanners. A spanner P crosses a list of thread-contiguous components C when P and C share at least one component and when it is the case that NOT ALL of the components in P are also in C. In other words, there is some intersection – but not total intersection – between the components of P and C.

Compare ‘crossing’ spanners with ‘covered’ spanners. Compare ‘crossing’ spanners with ‘dominant’ spanners. Compare ‘crossing’ spanners with ‘contained’ spanners. Compare ‘crossing’ spanners with ‘attached’ spanners. Changed in version 2.0: renamed `spannertools.get_crossing()` to `spannertools.get_spanners_that_cross_components()`.

spannertools.get_spanners_that_dominate_component_pair

spannertools.get_spanners_that_dominate_component_pair (*left, right*)

Return Python list of (spanner, index) pairs. ‘left’ must be either an Abjad component or None. ‘right’ must be either an Abjad component or None.

If both ‘left’ and ‘right’ are components, then ‘left’ and ‘right’ must be thread-contiguous.

This is a special version of `spannertools.get_spanners_that_dominate_components()`. This version is useful for finding spanners that dominate a zero-length ‘crack’ between components, as in `t[2:2]`. Changed in version 2.0: renamed `spannertools.get_dominant_between()` to `spannertools.get_spanners_that_dominate_component_pair()`.

spannertools.get_spanners_that_dominate_components

spannertools.get_spanners_that_dominate_components (*components*)

Return Python list of (spanner, index) pairs. Each (spanner, index) pair gives a spanner which dominates all components in ‘components’ together with the start-index at which spanner first encounters ‘components’.

Use this helper to ‘lift’ any and all spanners temporarily from ‘components’, perform some action to the underlying score tree, and then reattach all spanners to new score components.

This operation always leaves all expressions in tact. Changed in version 2.0: renamed `spannertools.get_dominant()` to `spannertools.get_spanners_that_dominate_components()`.

spannertools.get_spanners_that_dominate_container_components_from_to

spannertools.get_spanners_that_dominate_container_components_from_to (*container,*
start,
stop)

Return Python list of (spanner, index) pairs. Each spanner dominates the components specified by slice with start index ‘start’ and stop index ‘stop’. Generalization of dominant spanner-finding functions for slices. This exists for slices like `t[2:2]` that are empty lists. Changed in version 2.0: renamed `spannertools.get_dominant_slice()` to `spannertools.get_spanners_that_dominate_container_components_from_to()`.

spannertools.get_the_only_spanner_attached_to_any_improper_parent_of_component

spannertools.get_the_only_spanner_attached_to_any_improper_parent_of_component (*component*, *klass=None*)

New in version 1.1. Get the only spanner attached to any improper parent *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> trill = spannertools.TrillSpanner(staff)
>>> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

>>> print spannertools.get_the_only_spanner_attached_to_component(staff)
TrillSpanner({c'8, d'8, e'8, f'8})
```

Raise missing spanner error when no spanner attached to *component*.

Raise extra spanner error when more than one spanner attached to *component*.

Return a single spanner.

Note: function will usually be called with *klass* specifier set.

spannertools.get_the_only_spanner_attached_to_component

spannertools.get_the_only_spanner_attached_to_component (*component*, *klass=None*)

New in version 1.1. Get the only spanner attached to *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> slur = spannertools.SlurSpanner(staff.leaves)
>>> trill = spannertools.TrillSpanner(staff)
>>> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

>>> print spannertools.get_the_only_spanner_attached_to_component(staff)
TrillSpanner({c'8, d'8, e'8, f'8})
```

Raise missing spanner error when no spanner attached to *component*.

Raise extra spanner error when more than one spanner attached to *component*.

Return a single spanner.

Note: function will usually be called with *klass* specifier set.

`spannertools.is_component_with_spanner_attached`

`spannertools.is_component_with_spanner_attached` (*expr*, *klass=None*)

New in version 2.0. True when *expr* is a component with spanner attached:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(staff.leaves)
>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}

>>> spannertools.is_component_with_spanner_attached(staff[0])
True
```

Otherwise false:

```
>>> spannertools.is_component_with_spanner_attached(staff)
False
```

When *klass* is not none then true when *expr* is a component with a spanner of *klass* attached.

Return true or false.

`spannertools.iterate_components_in_spanner`

`spannertools.iterate_components_in_spanner` (*spanner*, *klass=None*, *reverse=False*)

New in version 2.10. Yield components in *spanner* one at a time from left to right:

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> p = beamtools.BeamSpanner(t[2:])

>>> notes = spannertools.iterate_components_in_spanner(p, klass=Note)

>>> for note in notes:
...     note
Note("e'8")
Note("f'8")
```

Yield components in *spanner* one at a time from right to left:

```
::

>>> notes = spannertools.iterate_components_in_spanner(p, klass=Note, reverse=True)

>>> for note in notes:
...     note
Note("f'8")
Note("e'8")
```

Return generator.

spannertools.make_covered_spanner_schema

`spannertools.make_covered_spanner_schema` (*components*)

New in version 2.0. Make schema of spanners covered by *components*:

```
>>> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 4)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(voice)
>>> beam = beamtools.BeamSpanner(voice.leaves[:4])
>>> slur = spannertools.SlurSpanner(voice[-2:])

>>> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
    d'8
  ]
  {
    e'8
    f'8 ]
  }
  {
    g'8 (
    a'8
  )
  {
    b'8
    c''8 )
  }
}

>>> spannertools.make_covered_spanner_schema([voice])
{BeamSpanner(c'8, d'8, e'8, f'8): [2, 3, 5, 6], SlurSpanner(|2/8(2)|, |2/8(2)|): [7, 10]}
```

Return dictionary.

spannertools.make_dynamic_spanner_below_with_nib_at_right

`spannertools.make_dynamic_spanner_below_with_nib_at_right` (*dynamic_text*, *components=None*)

New in version 2.0. Span *components* with text spanner. Position spanner below staff and configure with *dynamic_text*, solid line and upward-pointing nib at right.

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> spannertools.make_dynamic_spanner_below_with_nib_at_right('mp', t[:])
TextSpanner(c'8, d'8, e'8, f'8)
>>> f(t)
\new Staff {
  \override TextSpanner #'bound-details #'left #'text = \markup { \dynamic { mp } }
  \override TextSpanner #'bound-details #'right #'text = \markup { \draw-line #'(0 . 1) }
  \override TextSpanner #'bound-details #'right-broken #'text = ##f
  \override TextSpanner #'dash-fraction = #1
  \override TextSpanner #'direction = #down
  c'8 \startTextSpan
  d'8
  e'8
  f'8 \stopTextSpan
}
```

```

\revert TextSpanner #'bound-details #'left #'text
\revert TextSpanner #'bound-details #'right #'text
\revert TextSpanner #'bound-details #'right-broken #'text
\revert TextSpanner #'dash-fraction
\revert TextSpanner #'direction
}

```

Changed in version 2.0: renamed `spanners.dynamic_spanner_below_with_nib_at_right()` to `spannertools.make_dynamic_spanner_below_with_nib_at_right()`.

spannertools.make_solid_text_spanner_above_with_nib_at_right

`spannertools.make_solid_text_spanner_above_with_nib_at_right` (*left_text*, *components=None*)

New in version 2.0. Span *components* with text spanner. Position spanner above staff and configure with *left_text*, solid line and downward-pointing nib at right.

```

>>> t = Staff("c'8 d'8 e'8 f'8")
>>> spannertools.make_solid_text_spanner_above_with_nib_at_right('foo', t[:])
TextSpanner(c'8, d'8, e'8, f'8)

>>> f(t)
\new Staff {
  \override TextSpanner #'bound-details #'left #'text = \markup { foo }
  \override TextSpanner #'bound-details #'right #'text = \markup {
    \draw-line #'(0 . -1) }
  \override TextSpanner #'bound-details #'right-broken #'text = ##f
  \override TextSpanner #'dash-fraction = #1
  \override TextSpanner #'direction = #up
  c'8 \startTextSpan
  d'8
  e'8
  f'8 \stopTextSpan
  \revert TextSpanner #'bound-details #'left #'text
  \revert TextSpanner #'bound-details #'right #'text
  \revert TextSpanner #'bound-details #'right-broken #'text
  \revert TextSpanner #'dash-fraction
  \revert TextSpanner #'direction
}

```

Changed in version 2.0: renamed `spanners.solid_text_spanner_above_with_nib_at_right()` to `spannertools.make_solid_text_spanner_above_with_nib_at_right()`.

spannertools.make_solid_text_spanner_below_with_nib_at_right

`spannertools.make_solid_text_spanner_below_with_nib_at_right` (*left_text*, *components=None*)

New in version 2.0. Span *components* with text spanner. Position spanner below staff and configure with *left_text*, solid line and upward-pointing nib at right.

```

>>> t = Staff("c'8 d'8 e'8 f'8")
>>> spannertools.make_solid_text_spanner_below_with_nib_at_right('foo', t[:])
TextSpanner(c'8, d'8, e'8, f'8)
>>> f(t)
\new Staff {
  \override TextSpanner #'bound-details #'left #'text = \markup { foo }

```

```

\override TextSpanner #'bound-details #'right #'text = \markup { \draw-line #'(0 . 1) }
\override TextSpanner #'bound-details #'right-broken #'text = ##f
\override TextSpanner #'dash-fraction = #1
\override TextSpanner #'direction = #down
c'8 \startTextSpan
d'8
e'8
f'8 \stopTextSpan
\revert TextSpanner #'bound-details #'left #'text
\revert TextSpanner #'bound-details #'right #'text
\revert TextSpanner #'bound-details #'right-broken #'text
\revert TextSpanner #'dash-fraction
\revert TextSpanner #'direction
}

```

Changed in version 2.0: renamed `spanners.solid_text_spanner_below_with_nib_at_right()` to `spannertools.make_solid_text_spanner_below_with_nib_at_right()`.

spannertools.make_spanner_schema

`spannertools.make_spanner_schema` (*components*)

New in version 2.0. Make schema of spanners contained by *components*:

```

>>> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 4)
>>> pitchtools.set_ascending_named_diatonic_pitches_on_tie_chains_in_expr(voice)
>>> beam = beamtools.BeamSpanner(voice.leaves[:4])
>>> slur = spannertools.SlurSpanner(voice[-2:])

>>> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
    d'8
  ]
  {
    e'8
    f'8 ]
  }
  {
    g'8 (
    a'8
  )
  {
    b'8
    c''8 )
  }
}

>>> spannertools.make_spanner_schema(voice.leaves[2:4])
{BeamSpanner(c'8, d'8, e'8, f'8): [0, 1]}

```

Return dictionary.

`spannertools.move_spanners_from_component_to_children_of_component`

`spannertools.move_spanners_from_component_to_children_of_component` (*donor*)

Give spanners attaching directly to donor to recipients. Usual use is to give attached spanners from parent to children, which is a composer-safe operation. Changed in version 2.0: renamed `spannertools.give_attached_to_children()` to `spannertools.move_spanners_from_component_to_children_of_component()`.

`spannertools.report_format_contributions_of_improper_spanners`

`spannertools.report_format_contributions_of_improper_spanners` (*component*, *klass=None*)

New in version 1.1. Report as string format contributions of all spanners attached to improper parentage of *component*:

```
>>> staff = Staff("c'8 [ ( d'8 e'8 f'8 ] )")
>>> trill = spannertools.TrillSpanner(staff)

>>> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

>>> print spannertools.report_spanner_format_contributions(staff[0])
BeamSpanner
  _format_right_of_leaf
  [
SlurSpanner
  _format_right_of_leaf
  (
```

Return string.

`spannertools.report_spanner_format_contributions`

`spannertools.report_spanner_format_contributions` (*component*, *klass=None*)

New in version 1.1. Report as string format contributions of all spanners attached to *component*:

```
>>> staff = Staff("c'8 [ ( d'8 e'8 f'8 ] )")
>>> trill = spannertools.TrillSpanner(staff)

>>> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

>>> print spannertools.report_spanner_format_contributions(staff[0])
BeamSpanner
  _format_right_of_leaf
```

```
    [
    SlurSpanner
      _format_right_of_leaf
      (
```

Return string.

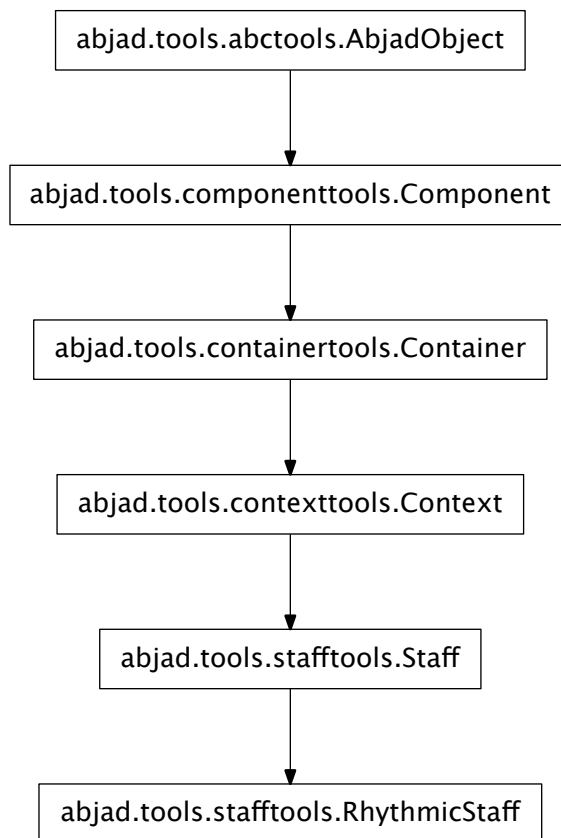
spannertools.withdraw_components_from_spanners_covered_by_components

spannertools.withdraw_components_from_spanners_covered_by_components (*components*)
Find every spanner covered by 'components'. Withdraw all components in 'components' from covered spanners. Return 'components'. The operation always leaves all score trees in tact. Changed in version 2.0: renamed `spannertools.withdraw_from_covered()` to `spannertools.withdraw_components_from_spanners_covered_by_components()`.

stafftools

concrete classes

stafftools.RhythmicStaff



class `stafftools.RhythmicStaff` (*music=None, context_name='RhythmicStaff', name=None*)
 Abjad model of a rhythmic staff.

Read-only Properties

`RhythmicStaff.contents_duration`
 Inherited from `containertools.Container`

`RhythmicStaff.duration_in_seconds`
 Inherited from `containertools.Container`

`RhythmicStaff.engraver_consists`
 New in version 2.0. Unordered set of LilyPond engravers to include in context definition.
 Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_consists.append('Horizontal_bracket_engraver')
>>> f(staff)
\new Staff \with {
  \consists Horizontal_bracket_engraver
} {
}
```

Inherited from `contexttools.Context`

RhythmicStaff.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_removals.append('Time_signature_engraver')
>>> f(staff)
\new Staff \with {
  \remove Time_signature_engraver
} {
}
```

Inherited from `contexttools.Context`

RhythmicStaff.is_semantic

Inherited from `contexttools.Context`

RhythmicStaff.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

RhythmicStaff.lilypond_format

Inherited from `contexttools.Context`

RhythmicStaff.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more components.

Inherited from `containertools.Container`

RhythmicStaff.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

RhythmicStaff.parent

Inherited from `componenttools.Component`

`RhythmicStaff.preprolated_duration`

Inherited from `containertools.Container`

`RhythmicStaff.prolated_duration`

Inherited from `componenttools.Component`

`RhythmicStaff.prolation`

Inherited from `componenttools.Component`

`RhythmicStaff.set`

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

`RhythmicStaff.spanners`

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`RhythmicStaff.start_offset`

Read-only start offset of component.

Inherited from `componenttools.Component`

`RhythmicStaff.start_offset_in_seconds`

Read-only start offset of comonent in seconds.

Inherited from `componenttools.Component`

`RhythmicStaff.stop_offset`

Read-only stop offset of component.

Inherited from `componenttools.Component`

`RhythmicStaff.stop_offset_in_seconds`

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`RhythmicStaff.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`RhythmicStaff.context_name`

Read / write name of context as a string.

Inherited from `contexttools.Context`

`RhythmicStaff.is_nonsemantic`

Set indicator of nonsemantic voice:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (5, 16), (5, 16)])
>>> voice = Voice(measures)
>>> voice.name = 'HiddenTimeSignatureVoice'

>>> voice.is_nonsemantic = True
```



```
>>> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    s1 * 5/16
  }
}
```

```
>>> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
>>> voice = Voice([])
```

```
>>> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Inherited from `contexttools.Context`

`RhythmicStaff.is_parallel`

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])
```

```
>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
>>> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True
```

```
>>> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Inherited from `containertools.Container`

`RhythmicStaff.name`

Read-write name of context. Must be string or none.

Inherited from `contexttools.Context`

Methods

`RhythmicStaff.append(component)`

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

`RhythmicStaff.extend(expr)`

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
```

```

    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```

Return none. New in version 2.3: `expr` may now be a LilyPond input string. Inherited from `containertools.Container`

`RhythmicStaff.index` (*component*)

Index *component* in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2

```

Return nonnegative integer.

Inherited from `containertools.Container`

`RhythmicStaff.insert` (*i*, *component*)

Insert *component* in container at index *i*:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}

```

Return none.

Inherited from `containertools.Container`

`RhythmicStaff.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

`RhythmicStaff.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

`RhythmicStaff.__add__(expr)`

Concatenate containers self and *expr*. The operation `c = a + b` returns a new `Container` *c* with the content of both

a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

`RhythmicStaff.__contains__(expr)`

True if `expr` is in container, otherwise False.

Inherited from `containertools.Container`

`RhythmicStaff.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

`RhythmicStaff.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`RhythmicStaff.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmicStaff.__getitem__(i)`

Return component at index i in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`RhythmicStaff.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`RhythmicStaff.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`RhythmicStaff.__imul__(total)`

Multiply contents of container 'total' times. Return multiplied container.

Inherited from `containertools.Container`

`RhythmicStaff.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmicStaff.__len__()`

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`RhythmicStaff.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RhythmicStaff.__mul__(n)`

Inherited from `componenttools.Component`

`RhythmicStaff.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`RhythmicStaff.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`RhythmicStaff.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Inherited from `contexttools.Context`

`RhythmicStaff.__rmul__(n)`

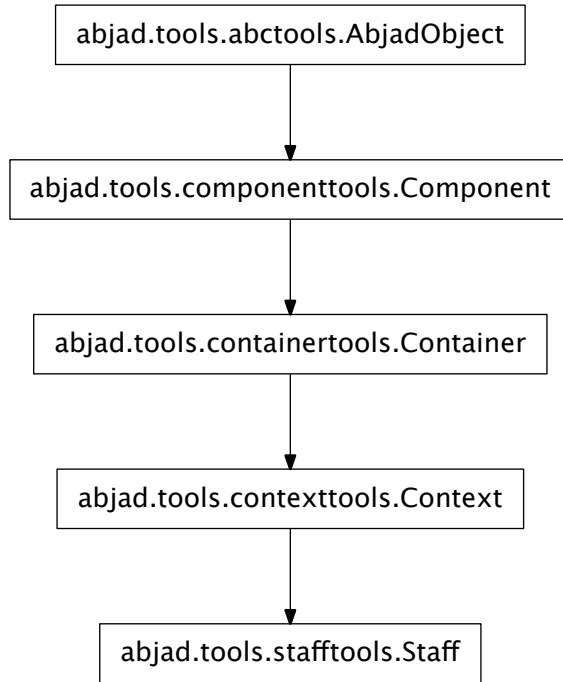
Inherited from `componenttools.Component`

`RhythmicStaff.__setitem__(i, expr)`

Set ‘`expr`’ in `self` at nonnegative integer index `i`. Or, set ‘`expr`’ in `self` at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with ‘`expr`’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

stafftools.Staff



class stafftools.**Staff** (*music=None, context_name='Staff', name=None*)
 Abjad model of a staff:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
  
```

Return staff object.

Read-only Properties

Staff.contents_duration

Inherited from containertools.Container

Staff.duration_in_seconds

Inherited from containertools.Container

Staff.engraver_consists

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_consists.append('Horizontal_bracket_engraver')
>>> f(staff)
\new Staff \with {
  \consists Horizontal_bracket_engraver
} {
}
```

Inherited from `contexttools.Context`

Staff.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_removals.append('Time_signature_engraver')
>>> f(staff)
\new Staff \with {
  \remove Time_signature_engraver
} {
}
```

Inherited from `contexttools.Context`

Staff.is_semantic

Inherited from `contexttools.Context`

Staff.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

Staff.lilypond_format

Inherited from `contexttools.Context`

Staff.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

Staff.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Staff.parent

Inherited from `componenttools.Component`

Staff.preprolated_duration

Inherited from `containertools.Container`

Staff.prolated_duration

Inherited from `componenttools.Component`

Staff.prolation

Inherited from `componenttools.Component`

Staff.set

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Staff.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Staff.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

Staff.start_offset_in_seconds

Read-only start offset of comonent in seconds.

Inherited from `componenttools.Component`

Staff.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

Staff.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Staff.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Staff.context_name

Read / write name of context as a string.

Inherited from `contexttools.Context`

Staff.is_nonsemantic

Set indicator of nonsemantic voice:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(
...     [(1, 8), (5, 16), (5, 16)])
>>> voice = Voice(measures)
>>> voice.name = 'HiddenTimeSignatureVoice'

>>> voice.is_nonsemantic = True
```

```
>>> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    s1 * 5/16
  }
}
```

```
>>> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
>>> voice = Voice([])
```

```
>>> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Inherited from `contexttools.Context`

Staff.is_parallel

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])
```

```
>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
>>> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True
```

```
>>> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Inherited from `containertools.Container`

Staff.name

Read-write name of context. Must be string or none.

Inherited from `contexttools.Context`

Methods

Staff.append(*component*)

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
>>> container.append(Note("f'8"))
```

```
>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

Staff.extend(*expr*)

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)
```

```
>>> f(container)
{
    c'8 [
    d'8
```

```

        e'8 ]
    }

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```

Return none. New in version 2.3: `expr` may now be a LilyPond input string. Inherited from `containertools.Container`

Staff.index(*component*)

Index *component* in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2

```

Return nonnegative integer.

Inherited from `containertools.Container`

Staff.insert(*i*, *component*)

Insert *component* in container at index *i*:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}

```

Return none.

Inherited from `containertools.Container`

`Staff.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

`Staff.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

`Staff.__add__(expr)`

Concatenate containers self and *expr*. The operation `c = a + b` returns a new `Container` *c* with the content of both

a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

Staff.**__contains__**(*expr*)

True if *expr* is in container, otherwise False.

Inherited from `containertools.Container`

Staff.**__delitem__**(*i*)

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

Staff.**__eq__**(*arg*)

True when `id(self) equals id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Staff.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Staff.**__getitem__**(*i*)

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

Staff.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Staff.**__iadd__**(*expr*)

`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

Staff.**__imul__**(*total*)

Multiply contents of container 'total' times. Return multiplied container.

Inherited from `containertools.Container`

Staff.**__le__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Staff.**__len__**()

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

Staff.**__lt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Staff.__mul__(n)`

Inherited from `componenttools.Component`

`Staff.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Staff.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`Staff.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Inherited from `contexttools.Context`

`Staff.__rmul__(n)`

Inherited from `componenttools.Component`

`Staff.__setitem__(i, expr)`

Set ‘`expr`’ in `self` at nonnegative integer index `i`. Or, set ‘`expr`’ in `self` at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with ‘`expr`’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

functions

`stafftools.all_are_staves`

`stafftools.all_are_staves(expr)`

New in version 2.6. True when `expr` is a sequence of Abjad staves:

```
>>> staff = Staff("c'4 d'4 e'4 f'4")
```

```
>>> stafftools.all_are_staves([staff])
True
```

True when `expr` is an empty sequence:

```
>>> stafftools.all_are_staves([])
True
```

Otherwise false:

```
>>> stafftools.all_are_staves('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

stafftools.get_first_staff_in_improper_parentage_of_component

`stafftools.get_first_staff_in_improper_parentage_of_component` (*component*)

New in version 2.0. Get first staff in improper parentage of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

>>> stafftools.get_first_staff_in_improper_parentage_of_component(staff[1])
Staff{4}
```

Return staff or none.

stafftools.get_first_staff_in_proper_parentage_of_component

`stafftools.get_first_staff_in_proper_parentage_of_component` (*component*)

New in version 2.0. Get first staff in proper parentage of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

>>> stafftools.get_first_staff_in_proper_parentage_of_component(staff[1])
Staff{4}
```

Return staff or none.

stafftools.make_rhythmic_sketch_staff

`stafftools.make_rhythmic_sketch_staff` (*music*)

Make rhythmic staff with transparent meter and transparent bar lines.

stringtools

functions

stringtools.arg_to_bidirectional_direction_string

`stringtools.arg_to_bidirectional_direction_string` (*arg*)

Convert *arg* to bidirectional direction string:


```
>>> from abjad.tools import stringtools

>>> stringtools.arg_to_bidirectional_direction_string('^')
'up'

>>> stringtools.arg_to_bidirectional_direction_string('_')
'down'

>>> stringtools.arg_to_bidirectional_direction_string(1)
'up'

>>> stringtools.arg_to_bidirectional_direction_string(-1)
'down'
```

If *arg* is 'up' or 'down', *arg* will be returned.

Return string or none.

stringtools.arg_to_bidirectional_lilypond_symbol

`stringtools.arg_to_bidirectional_lilypond_symbol(arg)`

Convert *arg* to bidirectional LilyPond symbol:

```
>>> from abjad.tools import stringtools

>>> stringtools.arg_to_tridirectional_lilypond_symbol(Up)
'^'

>>> stringtools.arg_to_tridirectional_lilypond_symbol(Down)
'_'

>>> stringtools.arg_to_tridirectional_lilypond_symbol(1)
'^'

>>> stringtools.arg_to_tridirectional_lilypond_symbol(-1)
'_'
```

If *arg* is '^' or '_', *arg* will be returned.

Return str or None.

stringtools.arg_to_tridirectional_direction_string

`stringtools.arg_to_tridirectional_direction_string(arg)`

Convert *arg* to tridirectional direction string:

```
>>> from abjad.tools import stringtools

>>> stringtools.arg_to_tridirectional_direction_string('^')
'up'

>>> stringtools.arg_to_tridirectional_direction_string('-')
'center'

>>> stringtools.arg_to_tridirectional_direction_string('_')
'down'
```

```
>>> stringtools.arg_to_tridirectional_direction_string(1)
'up'

>>> stringtools.arg_to_tridirectional_direction_string(0)
'center'

>>> stringtools.arg_to_tridirectional_direction_string(-1)
'down'

>>> stringtools.arg_to_tridirectional_direction_string('default')
'center'
```

If *arg* is None, None will be returned.

Return str or None.

stringtools.arg_to_tridirectional_lilypond_symbol

`stringtools.arg_to_tridirectional_lilypond_symbol(arg)`

Convert *arg* to tridirectional LilyPond symbol:

```
>>> from abjad.tools import stringtools

>>> stringtools.arg_to_tridirectional_lilypond_symbol(Up)
'^'

>>> stringtools.arg_to_tridirectional_lilypond_symbol('neutral')
'-'

>>> stringtools.arg_to_tridirectional_lilypond_symbol('default')
'_'

>>> stringtools.arg_to_tridirectional_lilypond_symbol(Down)
'_'

>>> stringtools.arg_to_tridirectional_lilypond_symbol(1)
'^'

>>> stringtools.arg_to_tridirectional_lilypond_symbol(0)
'-'

>>> stringtools.arg_to_tridirectional_lilypond_symbol(-1)
'_'
```

If *arg* is None, None will be returned.

If *arg* is '^', '-', or '_', *arg* will be returned.

Return string or None.

stringtools.arg_to_tridirectional_ordinal_constant

`stringtools.arg_to_tridirectional_ordinal_constant(arg)`

Convert *arg* to tridirectional ordinal constant:

```
>>> from abjad.tools import stringtools
```

```
>>> stringtools.arg_to_tridirectional_ordinal_constant('^')
Up

>>> stringtools.arg_to_tridirectional_ordinal_constant('_')
Down

>>> stringtools.arg_to_tridirectional_ordinal_constant(1)
Up

>>> stringtools.arg_to_tridirectional_ordinal_constant(-1)
Down
```

If *arg* is Up, Center or Down, *arg* will be returned.

Return OrdinalConstant or None.

stringtools.capitalize_string_start

stringtools.capitalize_string_start (*string*)

New in version 2.5. Capitalize *string*:

```
>>> string = 'violin I'

>>> stringtools.capitalize_string_start(string)
'Violin I'
```

Function differs from built-in `string.capitalize()`.

This function affects only `string[0]` and leaves noninitial characters as-is.

Built-in `string.capitalize()` forces noninitial characters to lowercase.

```
>>> string.capitalize()
'Violin i'
```

Return newly constructed string. Changed in version 2.9: renamed `iotools.capitalize_string_start()` to `stringtools.capitalize_string_start()`.

stringtools.format_input_lines_as_doc_string

stringtools.format_input_lines_as_doc_string (*input_lines*)

New in version 2.0. Format *input_lines* as doc string.

Format expressions intelligently.

Treat blank lines intelligently.

Capture hash-suffixed line output.

Use when writing docstrings.

Example skipped because docstring goes crazy on example input.

stringtools.format_input_lines_as_regression_test

stringtools.format_input_lines_as_regression_test (*input_lines*, *tab_width=3*)

New in version 2.0. Format *input_lines* as regression test:

```
>>> input_lines = '''
... staff = Staff("c'8 d'8 e'8 f'8")
... beamtools.BeamSpanner(staff.leaves)
... f(staff)
...
... tuplettools.FixedDurationTuplet(Duration(2, 8), staff[:3])
... f(staff)
... '''

>>> stringtools.format_input_lines_as_regression_test(input_lines)

staff = Staff("c'8 d'8 e'8 f'8")
beamtools.BeamSpanner(staff.leaves)

r'''
\new Staff {
  c'8 [
    d'8
    e'8
    f'8 ]
}
'''

tuplettools.FixedDurationTuplet(Duration(2, 8), staff[:3])

r'''
\new Staff {
  \times 2/3 {
    c'8 [
      d'8
      e'8
    ]
  }
  f'8 ]
}

assert componenttools.is_well_formed_component(staff)
assert staff.lilypond_format == "\\new Staff {
  \\n\\t\\times 2/3 {\\n\\t\\tc'8 [\\n\\t\\td'8\\n\\t\\te'8\\n\\t}\\n\\tf'8 ]\\n}"
'''
```

Format expressions intelligently.

Treat blank lines intelligently.

Remove line-final hash characters.

Used when writing tests.

stringtools.is_lowercamelcase_string

stringtools.is_lowercamelcase_string(*expr*)

New in version 2.5. True when *expr* is a string and is lowercamelcase:

```
>>> stringtools.is_lowercamelcase_string('fooBar')
True
```

False otherwise:

```
>>> stringtools.is_lowercamelcase_string('FooBar')
False
```

Return boolean. Changed in version 2.9: renamed `iotools.is_lowercamelcase_string()` to `stringtools.is_lowercamelcase_string()`.

stringtools.is_space_delimited_lowercase_string

`stringtools.is_space_delimited_lowercase_string(expr)`

New in version 2.5. True when *expr* is a string and is space-delimited lowercase:

```
>>> stringtools.is_space_delimited_lowercase_string('foo bar')
True
```

False otherwise:

```
>>> stringtools.is_space_delimited_lowercase_string('foo_bar')
False
```

Return boolean. Changed in version 2.9: renamed `iotools.is_space_delimited_lowercase_string()` to `stringtools.is_space_delimited_lowercase_string()`.

stringtools.is_underscore_delimited_lowercase_file_name

`stringtools.is_underscore_delimited_lowercase_file_name(expr)`

New in version 2.7. True when *expr* is a string and is underscore-delimited lowercase file name with extension:

```
>>> stringtools.is_underscore_delimited_lowercase_file_name('foo_bar')
True
```

False otherwise:

```
>>> stringtools.is_underscore_delimited_lowercase_file_name('foo.bar.blah')
False
```

Return boolean. Changed in version 2.9: renamed `iotools.is_underscore_delimited_lowercase_file_name()` to `stringtools.is_underscore_delimited_lowercase_file_name()`.

stringtools.is_underscore_delimited_lowercase_file_name_with_extension

`stringtools.is_underscore_delimited_lowercase_file_name_with_extension(expr)`

New in version 2.7. True when *expr* is a string and is underscore-delimited lowercase file name with extension:

```
>>> stringtools.is_underscore_delimited_lowercase_file_name_with_extension('foo_bar.blah')
True
```

False otherwise:

```
>>> stringtools.is_underscore_delimited_lowercase_file_name_with_extension('foo.bar.blah')
False
```

Return boolean. Changed in version 2.9: renamed `iotools.is_underscore_delimited_lowercase_file_name_w` to `stringtools.is_underscore_delimited_lowercase_file_name_with_extension()`.

stringtools.is_underscore_delimited_lowercase_package_name

stringtools.is_underscore_delimited_lowercase_package_name (*expr*)

New in version 2.5. True when *expr* is a string and is underscore-delimited lowercase package name:

```
>>> stringtools.is_underscore_delimited_lowercase_package_name('foo.bar.blah_package')
True
```

False otherwise:

```
>>> stringtools.is_underscore_delimited_lowercase_package_name('foo.bar.BlahPackage')
False
```

Return boolean. Changed in version 2.9: renamed `iotools.is_underscore_delimited_lowercase_package_name()` to `stringtools.is_underscore_delimited_lowercase_package_name()`.

stringtools.is_underscore_delimited_lowercase_string

stringtools.is_underscore_delimited_lowercase_string (*expr*)

New in version 2.5. True when *expr* is a string and is underscore delimited lowercase:

```
>>> stringtools.is_underscore_delimited_lowercase_string('foo_bar')
True
```

False otherwise:

```
>>> stringtools.is_underscore_delimited_lowercase_string('foo bar')
False
```

Return boolean. Changed in version 2.9: renamed `iotools.is_underscore_delimited_lowercase_string()` to `stringtools.is_underscore_delimited_lowercase_string()`.

stringtools.is_uppercamelcase_string

stringtools.is_uppercamelcase_string (*expr*)

New in version 2.5. True when *expr* is a string and is uppercamelcase:

```
>>> stringtools.is_uppercamelcase_string('FooBar')
True
```

False otherwise:

```
>>> stringtools.is_uppercamelcase_string('fooBar')
False
```

Return boolean. Changed in version 2.9: renamed `iotools.is_uppercamelcase_string()` to `stringtools.is_uppercamelcase_string()`.

stringtools.space_delimited_lowercase_to_uppercamelcase

stringtools.space_delimited_lowercase_to_uppercamelcase (*string*)

New in version 2.6. Change space-delimited lowercase *string* to uppercamelcase:

```
>>> string = 'bass figure alignment positioning'
>>> stringtools.space_delimited_lowercase_to_uppercamelcase(string)
'BassFigureAlignmentPositioning'
```

Return string. Changed in version 2.9: renamed `iotools.space_delimited_lowercase_to_uppercamelcase()` to `stringtools.space_delimited_lowercase_to_uppercamelcase()`.

stringtools.string_to_strict_directory_name

`stringtools.string_to_strict_directory_name(string)`

New in version 2.6. Change *string* to strict directory name:

```
>>> stringtools.string_to_strict_directory_name('Déja vu')
'deja_vu'
```

Strip accents from accented characters. Change all punctuation (including spaces) to underscore. Set to lower-case.

Return string. Changed in version 2.9: renamed `iotools.string_to_strict_directory_name()` to `stringtools.string_to_strict_directory_name()`.

stringtools.strip_diacritics_from_binary_string

`stringtools.strip_diacritics_from_binary_string(binary_string)`

New in version 2.5. Strip diacritics from *binary_string*:

```
>>> binary_string = 'Dvo\x5\x99\x3\xa1k'

>>> print binary_string
Dvořák

>>> stringtools.strip_diacritics_from_binary_string(binary_string)
'Dvorak'
```

Return ASCII string. Changed in version 2.9: renamed `iotools.strip_diacritics_from_binary_string()` to `stringtools.strip_diacritics_from_binary_string()`.

stringtools.underscore_delimited_lowercase_to_lowercamelcase

`stringtools.underscore_delimited_lowercase_to_lowercamelcase(string)`

New in version 2.0. Change underscore-delimited lowercase *string* to lowercamelcase:

```
>>> string = 'bass_figure_alignment_positioning'
>>> stringtools.underscore_delimited_lowercase_to_lowercamelcase(string)
'bassFigureAlignmentPositioning'
```

Changed in version 2.0: renamed `stringtools.underscore_delimited_lowercase_to_lowercamelcase()` to `stringtools.underscore_delimited_lowercase_to_lowercamelcase()`. Changed in version 2.9: renamed `iotools.underscore_delimited_lowercase_to_lowercamelcase()` to `stringtools.underscore_delimited_lowercase_to_lowercamelcase()`.

stringtools.underscore_delimited_lowercase_to_uppercamelcase

`stringtools.underscore_delimited_lowercase_to_uppercamelcase(string)`

New in version 2.0. Change underscore-delimited lowercase *string* to uppercamelcase:

```
>>> string = 'bass_figure_alignment_positioning'
>>> stringtools.underscore_delimited_lowercase_to_uppercamelcase(string)
'BassFigureAlignmentPositioning'
```

Changed in version 2.0: renamed `stringtools.underscore_delimited_lowercase_to_uppercamelcase()` to `stringtools.underscore_delimited_lowercase_to_uppercamelcase()`. Changed in version 2.9: renamed `iotools.underscore_delimited_lowercase_to_uppercamelcase()` to `stringtools.underscore_delimited_lowercase_to_uppercamelcase()`.

`stringtools.uppercamelcase_to_space_delimited_lowercase`

`stringtools.uppercamelcase_to_space_delimited_lowercase(string)`

New in version 2.6. Change uppercamelcase *string* to space-delimited lowercase:

```
>>> string = 'KeySignatureMark'

>>> stringtools.uppercamelcase_to_space_delimited_lowercase(string)
'key signature mark'
```

Return string. Changed in version 2.9: renamed `iotools.uppercamelcase_to_space_delimited_lowercase()` to `stringtools.uppercamelcase_to_space_delimited_lowercase()`.

`stringtools.uppercamelcase_to_underscore_delimited_lowercase`

`stringtools.uppercamelcase_to_underscore_delimited_lowercase(string)`

New in version 2.6. Change uppercamelcase *string* to underscore-delimited lowercase:

```
>>> string = 'KeySignatureMark'

>>> stringtools.uppercamelcase_to_underscore_delimited_lowercase(string)
'key_signature_mark'
```

Return string. Changed in version 2.9: renamed `iotools.uppercamelcase_to_underscore_delimited_lowercase()` to `stringtools.uppercamelcase_to_underscore_delimited_lowercase()`.

`tempotools`

functions

`tempotools.integer_tempo_to_multiplier_tempo_pairs`

`tempotools.integer_tempo_to_multiplier_tempo_pairs(integer_tempo, maxi-
mum_numerator=None, maxi-
mum_denominator=None)`

New in version 2.0. Return all multiplier, tempo pairs possible from *integer_tempo*.

Tempi must be no less than $\text{integer_tempo} / 2$ and not greater than $2 * \text{integer_tempo}$:

```
>>> from abjad.tools import tempotools

>>> pairs = tempotools.integer_tempo_to_multiplier_tempo_pairs(58, 8, 8)
>>> for pair in pairs:
...     pair
... 
```



```
(Fraction(1, 2), Fraction(29, 1))
(Fraction(1, 1), Fraction(58, 1))
(Fraction(3, 2), Fraction(87, 1))
(Fraction(2, 1), Fraction(116, 1))
```

Return list.

`tempotools.integer_tempo_to_multiplier_tempo_pairs_report`

`tempotools.integer_tempo_to_multiplier_tempo_pairs_report` (*integer_tempo*, *maximum_numerator=None*, *maximum_denominator=None*)

New in version 2.0. Print all multiplier, tempo pairs possible from *integer_tempo*.

Allow no tempi less than *integer_tempo* / 2 nor greater than 2 * *integer_tempo*:

```
>>> from abjad.tools import tempotools

>>> tempotools.integer_tempo_to_multiplier_tempo_pairs_report(58, 8, 8)
2:1      29
1:1      58
2:3      87
1:2     116
```

With more lenient numerator and denominator.

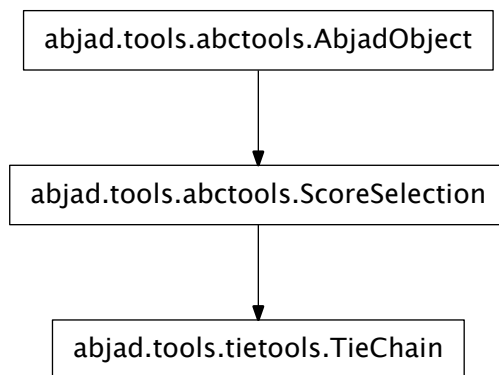
```
>>> tempotools.integer_tempo_to_multiplier_tempo_pairs_report(58, 30, 30)
2:1      29
29:15     30
29:16     32
29:17     34
29:18     36
29:19     38
29:20     40
29:21     42
29:22     44
29:23     46
29:24     48
29:25     50
29:26     52
29:27     54
29:28     56
1:1      58
29:30     60
2:3      87
1:2     116
```

Return none.

tietools

concrete classes

tietools.TieChain



class tietools.**TieChain** (*music*)

New in version 2.9. All the notes in a tie chain:

```
>>> staff = Staff("c' d' e' ~ e'")

>>> tietools.get_tie_chain(staff[2])
TieChain((Note("e'4"), Note("e'4")))
```

Tie chains are immutable score selections.

Read-only Properties

TieChain.all_leaves_are_in_same_parent

True when all leaves in tie chain are in same parent.

Return boolean.

TieChain.duration_in_seconds

Read-only duration in seconds of components in tie chain.

Return duration.

TieChain.head

Read-only reference to element 0 in tie chain.

TieChain.is_pitched

True when tie chain head is a note or chord.

Return boolean.

TieChain.is_trivial

True when length of tie chain is less than or equal to 1.

Return boolean.

TieChain.leaves

Read-only tuple of leaves in tie spanner.

TieChain.leaves_grouped_by_immediate_parents

Read-only list of leaves in tie chain grouped by immediate parents of leaves.

Return list of lists.

TieChain.music

Read-only tuple of components in selection.

Inherited from `abctools.ScoreSelection`

TieChain.preprolated_duration

Sum of preprolated durations of all components in tie chain.

TieChain.prolated_duration

Sum of prolated durations of all components in tie chain.

TieChain.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

TieChain.written_duration

Sum of written duration of all components in tie chain.

Special Methods

TieChain.__contains__ (*expr*)

Inherited from `abctools.ScoreSelection`

TieChain.__eq__ (*expr*)

Inherited from `abctools.ScoreSelection`

TieChain.__ge__ (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

TieChain.__getitem__ (*expr*)

Inherited from `abctools.ScoreSelection`

TieChain.__gt__ (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

TieChain.__le__ (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

TieChain.__len__ ()

Inherited from `abctools.ScoreSelection`

`TieChain.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TieChain.__ne__(expr)`

Inherited from `abctools.ScoreSelection`

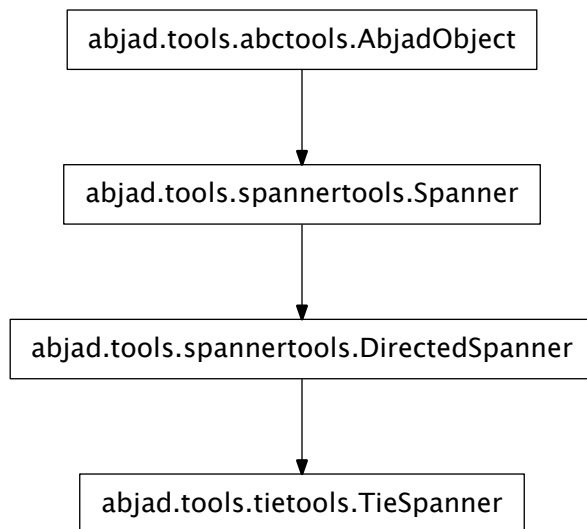
`TieChain.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`tietools.TieSpanner`



class `tietools.TieSpanner` (*music=None, direction=None*)

Abjad tie spanner:

```

>>> staff = Staff(notetools.make_repeated_notes(4))
>>> tietools.TieSpanner(staff[:])
TieSpanner(c'8, c'8, c'8, c'8)
  
```

```

>>> f(staff)
\new Staff {
  c'8 ~
  c'8 ~
  c'8 ~
  c'8
}
  
```

Return tie spanner.

Read-only Properties

`TieSpanner.components`

Return read-only tuple of components in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list. Inherited from `spannertools.Spanner`

`TieSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Inherited from `spannertools.Spanner`

`TieSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner-specific iteration tools.

Inherited from `spannertools.Spanner`

`TieSpanner.override`

LilyPond grob override component plug-in.

Inherited from `spannertools.Spanner`

`TieSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`TieSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Inherited from `spannertools.Spanner`

`TieSpanner.set`

LilyPond context setting component plug-in.

Inherited from `spannertools.Spanner`

`TieSpanner.start_offset`

Read-only start offset of spanner.

Inherited from `spannertools.Spanner`

`TieSpanner.stop_offset`

Read-only stop offset of spanner.

Inherited from `spannertools.Spanner`

`TieSpanner.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`TieSpanner.written_duration`
Sum of written duration of all components in spanner.

Inherited from `spannertools.Spanner`

Read/write Properties

`TieSpanner.direction`
Inherited from `spannertools.DirectedSpanner`

Methods

`TieSpanner.append(component)`
Add *component* to right of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)
```

```
>>> spanner.append(voice[2])
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TieSpanner.append_left(component)`
Add *component* to left of spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)
```

```
>>> spanner.append_left(voice[1])
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TieSpanner.clear()`
Remove all components from spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.clear()
>>> spanner
BeamSpanner()
```

Return none.

Inherited from `spannertools.Spanner`

`TieSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8)

>>> spanner.extend(voice[2:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TieSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.extend_left(voice[:2])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

Return none.

Inherited from `spannertools.Spanner`

`TieSpanner.fracture(i, direction=None)`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are Left, Right and None.

Return original, left and right spanners.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> beam = beamtools.BeamSpanner(voice[:])
>>> beam
BeamSpanner(c'8, d'8, e'8, f'8)

>>> beam.fracture(1, direction=Left)
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

>>> print voice.lilypond_format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Set *direction=None* to fracture on both left and right sides.

Return tuple.

Inherited from `spannertools.Spanner`

`TieSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> left_beam = beamtools.BeamSpanner(voice[:2])
>>> right_beam = beamtools.BeamSpanner(voice[2:])

>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

>>> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
>>> print voice.lilypond_format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Return list.

Inherited from `spannertools.Spanner`

`TieSpanner.index(component)`

Return nonnegative integer index of *component* in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[2:])
>>> spanner
BeamSpanner(e'8, f'8)

>>> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Inherited from `spannertools.Spanner`

`TieSpanner.pop()`

Remove and return rightmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)

>>> spanner.pop()
Note("f'8")
```



```
>>> spanner
BeamSpanner(c'8, d'8, e'8)
```

Return component.

Inherited from `spannertools.Spanner`

`TieSpanner.pop_left()`

Remove and return leftmost component in spanner:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> spanner = beamtools.BeamSpanner(voice[:])
>>> spanner
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
>>> spanner.pop_left()
Note("c'8")
```

```
>>> spanner
BeamSpanner(d'8, e'8, f'8)
```

Return component.

Inherited from `spannertools.Spanner`

Special Methods

`TieSpanner.__call__(expr)`

New in version 2.9. Call spanner on *expr* as a shortcut to extend spanner:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")

>>> beam = beamtools.BeamSpanner()
>>> beam(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

>>> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

The method is provided as an experimental way of unifying spanner and mark attachment syntax.

Return spanner.

Inherited from `spannertools.Spanner`

`TieSpanner.__contains__(expr)`

Inherited from `spannertools.Spanner`

`TieSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TieSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TieSpanner.__getitem__(expr)`

Inherited from `spannertools.Spanner`

`TieSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TieSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TieSpanner.__len__()`

Inherited from `spannertools.Spanner`

`TieSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Inherited from `spannertools.Spanner`

`TieSpanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TieSpanner.__repr__()`

Inherited from `spannertools.Spanner`

functions

`tietools.add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration`

`tietools.add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration(tie_chain, multiplier)`

Add or remove *tie_chain* notes to achieve scaled written duration:

```
>>> staff = Staff("c'8 [ ]")
```

```
>>> f(staff)
```

```
\new Staff {
  c'8 [ ]
}
```

```
>>> tie_chain = tietools.get_tie_chain(staff[0])
```

```
>>> tietools.add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration(
```

```
...     tie_chain, Fraction(5, 4))
```

```
TieChain((Note("c'8"), Note("c'32")))
```

```
>>> f(staff)
\new Staff {
    c'8 [ ~
    c'32 ]
}
```

Return *tie_chain*. Changed in version 2.0: renamed `tietools.duration_scale()` to `tietools.add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration()`.

tietools.add_or_remove_tie_chain_notes_to_achieve_written_duration

`tietools.add_or_remove_tie_chain_notes_to_achieve_written_duration`(*tie_chain*,
new_written_duration)

Add or remove *tie_chain* notes to achieve *written_duration*:

```
>>> staff = Staff("c'8 [ ]")

>>> f(staff)
\new Staff {
    c'8 [ ]
}

>>> tie_chain = tietools.get_tie_chain(staff[0])
>>> tietools.add_or_remove_tie_chain_notes_to_achieve_written_duration(
...     tie_chain, Duration(5, 32))
TieChain((Note("c'8"), Note("c'32")))

>>> f(staff)
\new Staff {
    c'8 [ ~
    c'32 ]
}
```

Return *tie_chain*. Changed in version 2.0: renamed `tietools.duration_change()` to `tietools.add_or_remove_tie_chain_notes_to_achieve_written_duration()`.

tietools.apply_tie_spanner_to_leaf_pair

`tietools.apply_tie_spanner_to_leaf_pair`(*left*, *right*)

Apply tie spanner to *left* leaf and *right* leaf:

```
>>> staff = Staff("c'8 ~ c' c' c'")

>>> f(staff)
\new Staff {
    c'8 ~
    c'8
    c'8
    c'8
}

>>> tietools.apply_tie_spanner_to_leaf_pair(staff[1], staff[2])

>>> f(staff)
\new Staff {
    c'8 ~
```

```

        c'8 ~
        c'8
        c'8
    }

```

Handle existing tie spanners intelligently.

Return `None`. Changed in version 2.0: renamed `tietools.span_leaf_pair()` to `tietools.apply_tie_spanner_to_leaf_pair()`.

tietools.are_components_in_same_tie_spanner

tietools.are_components_in_same_tie_spanner(*components*)

True when *components* are in same tie spanner:

```

>>> voice = Voice("c'8 ~ c' d' ~ d'")

>>> f(voice)
\new Voice {
    c'8 ~
    c'8
    d'8 ~
    d'8
}

>>> tietools.are_components_in_same_tie_spanner(voice[:2])
True

```

Otherwise false:

```

>>> tietools.are_components_in_same_tie_spanner(voice[1:3])
False

```

Return boolean. Changed in version 2.0: renamed `tietools.are_in_same_spanner()` to `tietools.are_components_in_same_tie_spanner()`.

tietools.get_nontrivial_tie_chains_masked_by_components

tietools.get_nontrivial_tie_chains_masked_by_components(*components*)

Get nontrivial tie chains masked by *components*:

```

>>> staff = Staff("c'8 ~ c'4 d'8 ~ d'4 e'4.")

>>> f(staff)
\new Staff {
    c'8 ~
    c'4
    d'8 ~
    d'4
    e'4.
}

```

Return only nontrivial tie chains:

```

>>> tietools.get_nontrivial_tie_chains_masked_by_components(staff.leaves)
[TieChain((Note("c'8"), Note("c'4"))), TieChain((Note("d'8"), Note("d'4")))]

```

Return ‘masked’ tie chains when only some notes of a tie chain are passed in:

```
>>> tietools.get_nontrivial_tie_chains_masked_by_components(staff.leaves[1:2])
[TieChain((Note("c'4"),))]
```

Return list of zero or more (possibly masked) tie chains. Changed in version 2.9: renamed `tietools.get_tie_chains_in_expr()` to `tietools.get_nontrivial_tie_chains_masked_by_components()`.

tietools.get_tie_chain

`tietools.get_tie_chain(component)`

New in version 2.0. Get tie chain from *component*:

```
>>> staff = Staff("c'8 ~ c' d'4")

>>> f(staff)
\new Staff {
    c'8 ~
    c'8
    d'4
}

>>> tietools.get_tie_chain(staff[0])
TieChain((Note("c'8"), Note("c'8")))
```

Return tie chain.

tietools.get_tie_spanner_attached_to_component

`tietools.get_tie_spanner_attached_to_component(component)`

New in version 2.10. Get the only tie spanner attached to *component*:

```
>>> staff = Staff("c'8 ~ c'8 d'4")

>>> f(staff)
\new Staff {
    c'8 ~
    c'8
    d'4
}

>>> tietools.get_tie_spanner_attached_to_component(staff[0])
TieSpanner(c'8, c'8)
```

Return tie spanner.

Raise missing spanner error when no tie spanner attached to *component*.

Raise extra spanner error when more than one tie spanner attached to *component*.

tietools.is_component_with_tie_spanner_attached

`tietools.is_component_with_tie_spanner_attached(expr)`

New in version 2.0. True when *expr* is component with tie spanner attached:

```
>>> staff = Staff("c'8 ~ c' ~ c' ~ c'")

>>> f(staff)
\new Staff {
    c'8 ~
    c'8 ~
    c'8 ~
    c'8
}

>>> tietools.is_component_with_tie_spanner_attached(staff[1])
True
```

Otherwise false:

```
>>> tietools.is_component_with_tie_spanner_attached(staff)
False
```

Return boolean.

tietools.iterate_nontrivial_tie_chains_in_expr

`tietools.iterate_nontrivial_tie_chains_in_expr(expr, reverse=False)`
 Iterate nontrivial tie chains forward in *expr*:

```
>>> staff = Staff(r"c'4 ~ \times 2/3 { c'16 d'8 } e'8 f'4 ~ f'16")

>>> f(staff)
\new Staff {
    c'4 ~
    \times 2/3 {
        c'16
        d'8
    }
    e'8
    f'4 ~
    f'16
}

>>> for x in tietools.iterate_nontrivial_tie_chains_in_expr(staff):
...     x
...
TieChain((Note("c'4"), Note("c'16")))
TieChain((Note("f'4"), Note("f'16")))
```

Iterate nontrivial tie chains backward in *expr*:

```
::

>>> for x in tietools.iterate_nontrivial_tie_chains_in_expr(staff, reverse=True):
...     x
...
TieChain((Note("f'4"), Note("f'16")))
TieChain((Note("c'4"), Note("c'16")))
```

Return generator.

tietools.iterate_pitched_tie_chains_in_expr

`tietools.iterate_pitched_tie_chains_in_expr(expr, reverse=False)`

New in version 2.10. Iterate pitched tie chains forward in *expr*:

```
>>> staff = Staff(r"c'4 ~ \times 2/3 { c'16 d'8 } e'8 r8 f'8 ~ f'16 r8.")

>>> f(staff)
\new Staff {
  c'4 ~
  \times 2/3 {
    c'16
    d'8
  }
  e'8
  r8
  f'8 ~
  f'16
  r8.
}

>>> for x in tietools.iterate_pitched_tie_chains_in_expr(staff):
...     x
...
TieChain((Note("c'4"), Note("c'16")))
TieChain((Note("d'8"),))
TieChain((Note("e'8"),))
TieChain((Note("f'8"), Note("f'16")))
```

Iterate pitched tie chains backward in *expr*:

```
::

>>> for x in tietools.iterate_pitched_tie_chains_in_expr(staff, reverse=True):
...     x
...
TieChain((Note("f'8"), Note("f'16")))
TieChain((Note("e'8"),))
TieChain((Note("d'8"),))
TieChain((Note("c'4"), Note("c'16")))
```

Tie chains are pitched if they comprise notes or chords.

Tie chains are not pitched if they comprise rests or skips.

Return generator.

tietools.iterate_tie_chains_in_expr

`tietools.iterate_tie_chains_in_expr(expr, reverse=False)`

Iterate tie chains forward in *expr*:

```
>>> staff = Staff(r"c'4 ~ \times 2/3 { c'16 d'8 } e'8 f'4 ~ f'16")

>>> f(staff)
\new Staff {
  c'4 ~
  \times 2/3 {
```

```

        c'16
        d'8
    }
    e'8
    f'4 ~
    f'16
}

>>> for x in tietools.iterate_tie_chains_in_expr(staff):
...     x
...
TieChain((Note("c'4"), Note("c'16")))
TieChain((Note("d'8"),))
TieChain((Note("e'8"),))
TieChain((Note("f'4"), Note("f'16")))

```

Iterate tie chains backward in *expr*:

```

::

>>> for x in tietools.iterate_tie_chains_in_expr(staff, reverse=True):
...     x
...
TieChain((Note("f'4"), Note("f'16")))
TieChain((Note("e'8"),))
TieChain((Note("d'8"),))
TieChain((Note("c'4"), Note("c'16")))

```

Return generator.

tietools.iterate_topmost_tie_chains_and_components_in_expr

tietools.iterate_topmost_tie_chains_and_components_in_expr(*expr*)

Iterate topmost tie chains and components forward in *expr*:

```

>>> string = r"c'8 ~ c'32 d'8 ~ d'32 \times 2/3 { e'8 f'8 g'8 } a'8 ~ a'32 b'8 ~ b'32"
>>> staff = Staff(string)

>>> f(staff)
\new Staff {
    c'8 ~
    c'32
    d'8 ~
    d'32
    \times 2/3 {
        e'8
        f'8
        g'8
    }
    a'8 ~
    a'32
    b'8 ~
    b'32
}

>>> for x in tietools.iterate_topmost_tie_chains_and_components_in_expr(staff):
...     x

```



```
...
TieChain((Note("c'8"), Note("c'32")))
TieChain((Note("d'8"), Note("d'32")))
Tuplet(2/3, [e'8, f'8, g'8])
TieChain((Note("a'8"), Note("a'32")))
TieChain((Note("b'8"), Note("b'32")))
```

Raise tie chain error on overlapping tie chains.

Return generator. Changed in version 2.0: renamed `iterate.chained_contents()` to `tietools.iterate_topmost_tie_chains_and_components_in_expr()`.

tietools.remove_nonfirst_leaves_in_tie_chain

tietools.remove_nonfirst_leaves_in_tie_chain(*tie_chain*)

Remove nonfirst leaves in *tie_chain*:

```
>>> staff = Staff("c'4 ~ c'16")

>>> f(staff)
\new Staff {
    c'4 ~
    c'16
}

>>> tietools.remove_nonfirst_leaves_in_tie_chain(tietools.get_tie_chain(staff[0]))
TieChain((Note("c'4"),))

>>> f(staff)
\new Staff {
    c'4
}
```

Return *tie_chain*. Changed in version 2.0: renamed `tietools.truncate()` to `tietools.remove_all_leaves_in_tie_chain_except_first()`. Changed in version 2.9: renamed `tietools.remove_all_leaves_in_tie_chain_except_first()` to `tietools.remove_nonfirst_leaves_in_tie_chain()`.

tietools.remove_tie_spanners_from_components_in_expr

tietools.remove_tie_spanners_from_components_in_expr(*expr*)

Remove tie spanners components in *expr*:

```
>>> staff = Staff("c'4 ~ c'16 d'4 ~ d'16")

>>> f(staff)
\new Staff {
    c'4 ~
    c'16
    d'4 ~
    d'16
}

>>> tietools.remove_tie_spanners_from_components_in_expr(staff[:])
[Note("c'4"), Note("c'16"), Note("d'4"), Note("d'16")]
```

```
>>> f(staff)
\new Staff {
  c'4
  c'16
  d'4
  d'16
}
```

Return *expr*. Changed in version 2.0: renamed `componenttools.untie_shallow()` to `tietools.remove_tie_spanners_from_components_in_expr()`.

tietools.tie_chain_to_tuplet_with_proportions

`tietools.tie_chain_to_tuplet_with_proportions(chain, divisions, diminution=True, dotted=True)`

New in version 2.0. Generalized tie-chain division function.

Example 1a. Change *tie_chain* to augmented tuplet with proportions [1]. Avoid dots:

```
>>> staff = Staff("c'8 [ ~ c'16 c'16 ]")

>>> f(staff)
\new Staff {
  c'8 [ ~
  c'16
  c'16 ]
}

>>> tie_chain = tietools.get_tie_chain(staff[0])
>>> tietools.tie_chain_to_tuplet_with_proportions(
...     tie_chain, [1], diminution=False, dotted=False)
FixedDurationTuplet(3/16, [c'8])

>>> f(staff)
\new Staff {
  \fraction \times 3/2 {
    c'8 [
  ]
  c'16 ]
}
```

Exempl 1b. Change *tie_chain* to augment tuplet with proportions [1, 2]. Avoid dots:

```
>>> staff = Staff("c'8 [ ~ c'16 c'16 ]")

>>> f(staff)
\new Staff {
  c'8 [ ~
  c'16
  c'16 ]
}

>>> tie_chain = tietools.get_tie_chain(staff[0])
>>> tietools.tie_chain_to_tuplet_with_proportions(
...     tie_chain, [1, 2], diminution=False, dotted=False)
FixedDurationTuplet(3/16, [c'16, c'8])
```

```
>>> f(staff)
\new Staff {
  {
    c'16 [
      c'8
    ]
  }
  c'16 ]
}
```

Example 1c. Change *tie_chain* to augmented tuplet with proportions [1, 2, 2]. Avoid dots:

```
>>> staff = Staff("c'8 [ ~ c'16 c'16 ]")

>>> f(staff)
\new Staff {
  c'8 [ ~
    c'16
    c'16 ]
}

>>> tie_chain = tietools.get_tie_chain(staff[0])
>>> tietools.tie_chain_to_tuplet_with_proportions(
...     tie_chain, [1, 2, 2], diminution=False, dotted=False)
FixedDurationTuplet(3/16, [c'32, c'16, c'16])

>>> f(staff)
\new Staff {
  \fraction \times 6/5 {
    c'32 [
      c'16
      c'16
    ]
  }
  c'16 ]
}
```

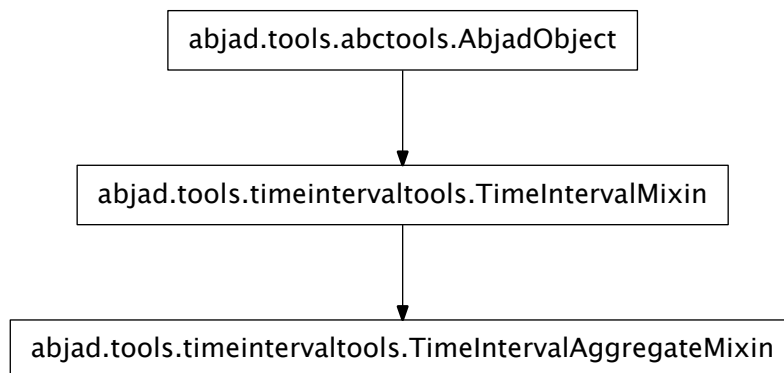
Todo

move to tuplettools.

`timeintervaltools`

abstract classes

`timeintervaltools.TimeIntervalAggregateMixin`



```
class timeintervaltools.TimeIntervalAggregateMixin(*args, **kwargs)
```

Read-only Properties

`TimeIntervalAggregateMixin.bounds`

Start and stop of self returned as `TimeInterval` instance:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> bounds = interval.bounds
>>> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
>>> bounds == interval
True
>>> bounds is interval
False
```

Returns *TimeInterval* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.center`

Center offset of start and stop offsets:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.center
Offset(6, 1)
```

Returns *Offset* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.duration`

Duration of the time interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.duration
Duration(8, 1)
```

Returns *Duration* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.earliest_start`

`TimeIntervalAggregateMixin.earliest_stop`

`TimeIntervalAggregateMixin.intervals`

`TimeIntervalAggregateMixin.latest_start`

`TimeIntervalAggregateMixin.latest_stop`

`TimeIntervalAggregateMixin.offset_counts`

`TimeIntervalAggregateMixin.offsets`

`TimeIntervalAggregateMixin.signature`

Tuple of start bound and stop bound.

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.signature
(Offset(2, 1), Offset(10, 1))
```

Returns 2-tuple of *Offset* instances.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.start`

Starting offset of interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.start
Offset(2, 1)
```

Returns *Offset* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.stop`

Stopping offset of interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.stop
Offset(10, 1)
```

Returns *Offset* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TimeIntervalAggregateMixin.find_intervals_intersecting_or_tangent_to_interval()`

`TimeIntervalAggregateMixin.find_intervals_intersecting_or_tangent_to_offset()`

`TimeIntervalAggregateMixin.find_intervals_starting_after_offset()`

`TimeIntervalAggregateMixin.find_intervals_starting_and_stopping_within_interval()`

`TimeIntervalAggregateMixin.find_intervals_starting_at_offset()`

`TimeIntervalAggregateMixin.find_intervals_starting_before_offset()`

`TimeIntervalAggregateMixin.find_intervals_starting_or_stopping_at_offset()`

`TimeIntervalAggregateMixin.find_intervals_starting_within_interval()`

`TimeIntervalAggregateMixin.find_intervals_stopping_after_offset()`

`TimeIntervalAggregateMixin.find_intervals_stopping_at_offset()`

`TimeIntervalAggregateMixin.find_intervals_stopping_before_offset()`

`TimeIntervalAggregateMixin.find_intervals_stopping_within_interval()`

`TimeIntervalAggregateMixin.get_overlap_with_interval(interval)`

Return amount of overlap with *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.is_contained_by_interval(interval)`

True if interval is contained by *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.is_container_of_interval(interval)`

True if interval contains *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.is_overlapped_by_interval(interval)`

True if interval is overlapped by *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.is_tangent_to_interval(interval)`

True if interval is tangent to *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.quantize_to_rational(rational)`

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.scale_by_rational(rational)`

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.scale_to_rational(rational)`

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.shift_by_rational(rational)`

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.shift_to_rational` (*rational*)

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.split_at_rationals` (**rationals*)

Inherited from `timeintervaltools.TimeIntervalMixin`

Special Methods

`TimeIntervalAggregateMixin.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TimeIntervalAggregateMixin.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimeIntervalAggregateMixin.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TimeIntervalAggregateMixin.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimeIntervalAggregateMixin.__lt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimeIntervalAggregateMixin.__ne__` (*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TimeIntervalAggregateMixin.__nonzero__` ()

Inherited from `timeintervaltools.TimeIntervalMixin`

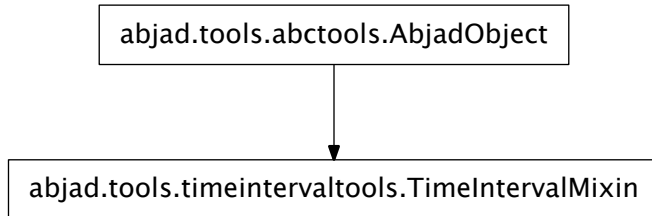
`TimeIntervalAggregateMixin.__repr__` ()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

timeintervaltools.TimeIntervalMixin



```
class timeintervaltools.TimeIntervalMixin(*args, **kwargs)
```

Read-only Properties

`TimeIntervalMixin.bounds`

Start and stop of self returned as `TimeInterval` instance:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> bounds = interval.bounds
>>> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
>>> bounds == interval
True
>>> bounds is interval
False
```

Returns *TimeInterval* instance.

`TimeIntervalMixin.center`

Center offset of start and stop offsets:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.center
Offset(6, 1)
```

Returns *Offset* instance.

`TimeIntervalMixin.duration`

Duration of the time interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.duration
Duration(8, 1)
```

Returns *Duration* instance.

`TimeIntervalMixin.signature`

Tuple of start bound and stop bound.


```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.signature
(Offset(2, 1), Offset(10, 1))
```

Returns 2-tuple of *Offset* instances.

`TimeIntervalMixin.start`

Starting offset of interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.start
Offset(2, 1)
```

Returns *Offset* instance.

`TimeIntervalMixin.stop`

Stopping offset of interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.stop
Offset(10, 1)
```

Returns *Offset* instance.

`TimeIntervalMixin.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TimeIntervalMixin.get_overlap_with_interval(interval)`

Return amount of overlap with *interval*.

`TimeIntervalMixin.is_contained_by_interval(interval)`

True if *interval* is contained by *interval*.

`TimeIntervalMixin.is_container_of_interval(interval)`

True if *interval* contains *interval*.

`TimeIntervalMixin.is_overlapped_by_interval(interval)`

True if *interval* is overlapped by *interval*.

`TimeIntervalMixin.is_tangent_to_interval(interval)`

True if *interval* is tangent to *interval*.

`TimeIntervalMixin.quantize_to_rational(rational)`

`TimeIntervalMixin.scale_by_rational(rational)`

`TimeIntervalMixin.scale_to_rational(rational)`

`TimeIntervalMixin.shift_by_rational(rational)`

`TimeIntervalMixin.shift_to_rational(rational)`

`TimeIntervalMixin.split_at_rationals(*rationals)`

Special Methods

`TimeIntervalMixin.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TimeIntervalMixin.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimeIntervalMixin.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TimeIntervalMixin.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimeIntervalMixin.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimeIntervalMixin.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TimeIntervalMixin.__nonzero__()`

`TimeIntervalMixin.__repr__()`

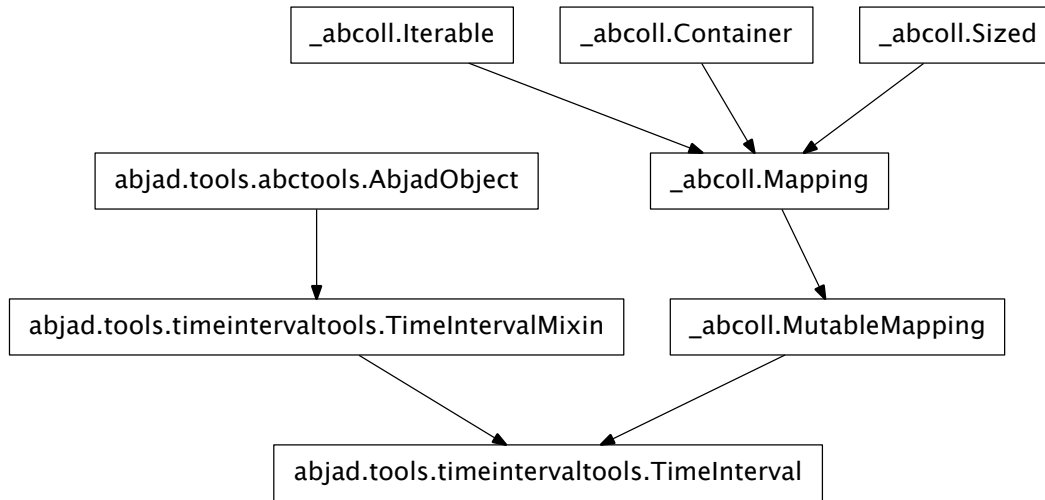
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

concrete classes

timeintervaltools.TimeInterval



class `timeintervaltools.TimeInterval` (*args)
 A start / stop pair, carrying some metadata.

Read-only Properties

`TimeInterval.bounds`

Start and stop of self returned as `TimeInterval` instance:

```

>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> bounds = interval.bounds
>>> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
>>> bounds == interval
True
>>> bounds is interval
False
  
```

Returns *TimeInterval* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.center`

Center point of start and stop bounds.

`TimeInterval.duration`

stop bound minus start bound.

`TimeInterval.signature`

Tuple of start bound and stop bound.

`TimeInterval.start`
start bound.

`TimeInterval.stop`
stop bound.

`TimeInterval.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TimeInterval.clear()`
Inherited from `_abcoll.MutableMapping`
`TimeInterval.get(key, default=None)`
Inherited from `_abcoll.Mapping`

`TimeInterval.get_overlap_with_interval(interval)`
Return amount of overlap with *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.is_contained_by_interval(interval)`
True if interval is contained by *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.is_container_of_interval(interval)`
True if interval contains *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.is_overlapped_by_interval(interval)`
True if interval is overlapped by *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.is_tangent_to_interval(interval)`
True if interval is tangent to *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.items()`
Inherited from `_abcoll.Mapping`

`TimeInterval.iteritems()`
Inherited from `_abcoll.Mapping`

`TimeInterval.iterkeys()`
Inherited from `_abcoll.Mapping`

`TimeInterval.intervalvalues()`
Inherited from `_abcoll.Mapping`

`TimeInterval.keys()`
Inherited from `_abcoll.Mapping`

`TimeInterval.pop(key, default=<object object at 0x1002b0040>)`
Inherited from `_abcoll.MutableMapping`

`TimeInterval.popitem()`
Inherited from `_abcoll.MutableMapping`

`TimeInterval.quantize_to_rational(rational)`
`TimeInterval.scale_by_rational(rational)`
`TimeInterval.scale_to_rational(rational)`
`TimeInterval.setdefault(key, default=None)`
 Inherited from `_abcoll.MutableMapping`
`TimeInterval.shift_by_rational(rational)`
`TimeInterval.shift_to_rational(rational)`
`TimeInterval.split_at_rationals(*rationals)`
`TimeInterval.update(*args, **kws)`
 Inherited from `_abcoll.MutableMapping`
`TimeInterval.values()`
 Inherited from `_abcoll.Mapping`

Special Methods

`TimeInterval.__contains__(key)`
 Inherited from `_abcoll.Mapping`
`TimeInterval.__delitem__(item)`
`TimeInterval.__eq__(other)`
`TimeInterval.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`
`TimeInterval.__getitem__(item)`
`TimeInterval.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`
`TimeInterval.__hash__()`
`TimeInterval.__iter__()`
`TimeInterval.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`
`TimeInterval.__len__()`
`TimeInterval.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`
`TimeInterval.__ne__(other)`

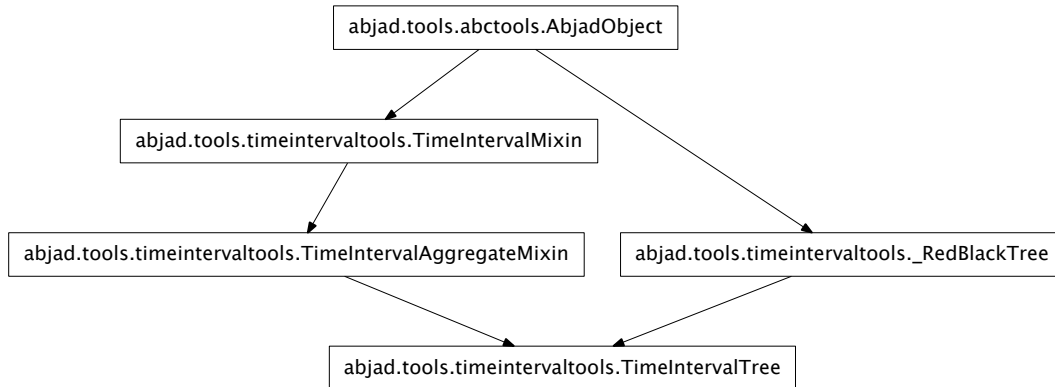
`TimeInterval.__nonzero__()`

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.__repr__()`

`TimeInterval.__setitem__(item, value)`

`timeintervaltools.TimeIntervalTree`



class `timeintervaltools.TimeIntervalTree` (*intervals=None*)

An augmented red-black tree for storing and searching for intervals of time (rather than pitch).

This allows for the arbitrary placement of blocks of material along a time-line. While this functionality could be achieved with Python's built-in collections, this class reduces the complexity of the search process, such as locating overlapping intervals.

`TimeIntervalTrees` can be instantiated without contents, or from a mixed collection of other `TimeIntervalTrees` and / or `TimeIntervals`. The input will be parsed recursively:

```

>>> from abjad.tools.timeintervaltools import TimeIntervalTree
>>> from abjad.tools.timeintervaltools import TimeInterval

>>> interval_one = TimeInterval(0, 10)
>>> interval_two = TimeInterval(1, 8)
>>> interval_three = TimeInterval(3, 13)
>>> tree = TimeIntervalTree([interval_one, interval_two, interval_three])

>>> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(10, 1), {}),
    TimeInterval(Offset(1, 1), Offset(8, 1), {}),
    TimeInterval(Offset(3, 1), Offset(13, 1), {})
])
  
```

Return *TimeIntervalTree* instance.

Read-only Properties

TimeIntervalTree.bounds

Start and stop of self returned as *TimeInterval* instance:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> bounds = interval.bounds
>>> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
>>> bounds == interval
True
>>> bounds is interval
False
```

Returns *TimeInterval* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTree.center

Center offset of start and stop offsets:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.center
Offset(6, 1)
```

Returns *Offset* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTree.duration

Absolute difference of the stop and start values of the tree:

```
>>> from abjad.tools.timeintervaltools import *
>>> ti1 = TimeInterval(1, 2)
>>> ti2 = TimeInterval(3, (7, 2))
>>> tree = TimeIntervalTree([ti1, ti2])
>>> tree.duration
Duration(5, 2)
```

Empty trees have a duration of 0.

Return *Duration* instance.

TimeIntervalTree.earliest_start

The minimum start value of all intervals in the tree:

```
>>> from abjad.tools.timeintervaltools import *
>>> ti1 = TimeInterval(1, 2)
>>> ti2 = TimeInterval(3, (7, 2))
>>> tree = TimeIntervalTree([ti1, ti2])
>>> tree.earliest_start
Offset(1, 1)
```

Return *Offset* instance, or *None* if tree is empty.

TimeIntervalTree.earliest_stop

The minimum stop value of all intervals in the tree:

```
>>> from abjad.tools.timeintervaltools import *
>>> ti1 = TimeInterval(1, 2)
>>> ti2 = TimeInterval(3, (7, 2))
>>> tree = TimeIntervalTree([ti1, ti2])
```

```
>>> tree.earliest_stop
Offset(2, 1)
```

Return *Offset* instance, or *None* if tree is empty.

`TimeIntervalTree.intervals`

`TimeIntervalTree.latest_start`

The maximum start value of all intervals in the tree:

```
>>> from abjad.tools.timeintervaltools import *
>>> ti1 = TimeInterval(1, 2)
>>> ti2 = TimeInterval(3, (7, 2))
>>> tree = TimeIntervalTree([ti1, ti2])
>>> tree.latest_start
Offset(3, 1)
```

Return *Offset* instance, or *None* if tree is empty.

`TimeIntervalTree.latest_stop`

The maximum stop value of all intervals in the tree:

```
>>> from abjad.tools.timeintervaltools import *
>>> ti1 = TimeInterval(1, 2)
>>> ti2 = TimeInterval(3, (7, 2))
>>> tree = TimeIntervalTree([ti1, ti2])
>>> tree.latest_stop
Offset(7, 2)
```

Return *Offset* instance, or *None* if tree is empty.

`TimeIntervalTree.offset_counts`

Inherited from `timeintervaltools.TimeIntervalAggregateMixin`

`TimeIntervalTree.offsets`

Inherited from `timeintervaltools.TimeIntervalAggregateMixin`

`TimeIntervalTree.signature`

Tuple of start bound and stop bound.

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.signature
(Offset(2, 1), Offset(10, 1))
```

Returns 2-tuple of *Offset* instances.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.start`

Starting offset of interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.start
Offset(2, 1)
```

Returns *Offset* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.stop`

Stopping offset of interval:


```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.stop
Offset(10, 1)
```

Returns *Offset* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TimeIntervalTree.find_intervals_intersecting_or_tangent_to_interval(*args)`

Find all intervals in tree intersecting or tangent to the interval defined in *args*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> interval = TimeInterval(0, 1)
>>> found = tree.find_intervals_intersecting_or_tangent_to_interval(interval)
>>> sorted([x['name'] for x in found])
['a', 'b', 'c']

>>> interval = TimeInterval(3, 4)
>>> found = tree.find_intervals_intersecting_or_tangent_to_interval(interval)
>>> sorted([x['name'] for x in found])
['c', 'd']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_intersecting_or_tangent_to_offset(offset)`

Find all intervals in tree intersecting or tangent to *offset*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> offset = 1
>>> found = tree.find_intervals_intersecting_or_tangent_to_offset(offset)
>>> sorted([x['name'] for x in found])
['a', 'b', 'c']

>>> offset = 3
>>> found = tree.find_intervals_intersecting_or_tangent_to_offset(offset)
>>> sorted([x['name'] for x in found])
['c', 'd']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_starting_after_offset(offset)`

Find all intervals in tree starting after *offset*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> offset = 0
>>> found = tree.find_intervals_starting_after_offset(offset)
>>> sorted([x['name'] for x in found])
['b', 'd']

>>> offset = 1
>>> found = tree.find_intervals_starting_after_offset(offset)
>>> sorted([x['name'] for x in found])
['d']
```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_starting_and_stopping_within_interval (*args)
Find all intervals in tree starting and stopping within the interval defined by *args*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> interval = TimeInterval(1, 3)
>>> found = tree.find_intervals_starting_and_stopping_within_interval(interval)
>>> sorted([x['name'] for x in found])
['b', 'd']

>>> interval = TimeInterval(-1, 2)
>>> found = tree.find_intervals_starting_and_stopping_within_interval(interval)
>>> sorted([x['name'] for x in found])
['a', 'b']
```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_starting_at_offset (offset)
Find all intervals in tree starting at *offset*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> offset = 0
>>> found = tree.find_intervals_starting_at_offset(offset)
>>> sorted([x['name'] for x in found])
['a', 'c']

>>> offset = 1
>>> found = tree.find_intervals_starting_at_offset(offset)
>>> sorted([x['name'] for x in found])
['b']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_starting_before_offset(offset)`

Find all intervals in tree starting before *offset*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> offset = 1
>>> found = tree.find_intervals_starting_before_offset(offset)
>>> sorted([x['name'] for x in found])
['a', 'c']

>>> offset = 2
>>> found = tree.find_intervals_starting_before_offset(offset)
>>> sorted([x['name'] for x in found])
['a', 'b', 'c']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_starting_or_stopping_at_offset(offset)`

Find all intervals in tree starting or stopping at *offset*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> offset = 2
>>> found = tree.find_intervals_starting_or_stopping_at_offset(offset)
>>> sorted([x['name'] for x in found])
['b', 'd']

>>> offset = 1
>>> found = tree.find_intervals_starting_or_stopping_at_offset(offset)
>>> sorted([x['name'] for x in found])
['a', 'b']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_starting_within_interval(*args)`

Find all intervals in tree starting within the interval defined by *args*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> interval = TimeInterval((-1, 2), (1, 2))
>>> found = tree.find_intervals_starting_within_interval(interval)
>>> sorted([x['name'] for x in found])
['a', 'c']

>>> interval = TimeInterval((1, 2), (5, 2))
>>> found = tree.find_intervals_starting_within_interval(interval)
>>> sorted([x['name'] for x in found])
['b', 'd']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_stopping_after_offset(offset)`

Find all intervals in tree stopping after *offset*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> offset = 1
>>> found = tree.find_intervals_stopping_after_offset(offset)
>>> sorted([x['name'] for x in found])
['b', 'c', 'd']

>>> offset = 2
>>> found = tree.find_intervals_stopping_after_offset(offset)
>>> sorted([x['name'] for x in found])
['c', 'd']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_stopping_at_offset(offset)`

Find all intervals in tree stopping at *offset*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> offset = 3
>>> found = tree.find_intervals_stopping_at_offset(offset)
>>> sorted([x['name'] for x in found])
['c', 'd']

>>> offset = 1
>>> found = tree.find_intervals_stopping_at_offset(offset)
>>> sorted([x['name'] for x in found])
['a']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_stopping_before_offset(offset)`

Find all intervals in tree stopping before *offset*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> offset = 3
>>> found = tree.find_intervals_stopping_before_offset(offset)
>>> sorted([x['name'] for x in found])
['a', 'b']

>>> offset = (7, 2)
>>> found = tree.find_intervals_stopping_before_offset(offset)
```

```
>>> sorted([x['name'] for x in found])
['a', 'b', 'c', 'd']
```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_stopping_within_interval(*args)

Find all intervals in tree stopping within the interval defined by *args*:

```
>>> a = TimeInterval(0, 1, {'name': 'a'})
>>> b = TimeInterval(1, 2, {'name': 'b'})
>>> c = TimeInterval(0, 3, {'name': 'c'})
>>> d = TimeInterval(2, 3, {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])

>>> interval = TimeInterval((3, 2), (5, 2))
>>> found = tree.find_intervals_stopping_within_interval(interval)
>>> sorted([x['name'] for x in found])
['b']

>>> interval = TimeInterval((5, 2), (7, 2))
>>> found = tree.find_intervals_stopping_within_interval(interval)
>>> sorted([x['name'] for x in found])
['c', 'd']
```

Return *TimeIntervalTree* instance.

TimeIntervalTree.get_overlap_with_interval(interval)

Return amount of overlap with *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTree.is_contained_by_interval(interval)

True if *interval* is contained by *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTree.is_container_of_interval(interval)

True if *interval* contains *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTree.is_overlapped_by_interval(interval)

True if *interval* is overlapped by *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTree.is_tangent_to_interval(interval)

True if *interval* is tangent to *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTree.quantize_to_rational(rational)

Quantize all intervals in tree to a multiple (1 or more) of *rational*:

```
>>> a = TimeInterval((1, 16), (1, 8), {'name': 'a'})
>>> b = TimeInterval((2, 7), (13, 7), {'name': 'b'})
>>> c = TimeInterval((3, 5), (8, 5), {'name': 'c'})
>>> d = TimeInterval((2, 3), (5, 3), {'name': 'd'})
>>> tree = TimeIntervalTree([a, b, c, d])
>>> tree
TimeIntervalTree([
  TimeInterval(Offset(1, 16), Offset(1, 8), {'name': 'a'}),
```

```

    TimeInterval(Offset(2, 7), Offset(13, 7), {'name': 'b'}),
    TimeInterval(Offset(3, 5), Offset(8, 5), {'name': 'c'}),
    TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'd'})
])

>>> rational = (1, 4)
>>> tree.quantize_to_rational(rational)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 4), {'name': 'a'}),
    TimeInterval(Offset(1, 4), Offset(7, 4), {'name': 'b'}),
    TimeInterval(Offset(1, 2), Offset(3, 2), {'name': 'c'}),
    TimeInterval(Offset(3, 4), Offset(7, 4), {'name': 'd'})
])

>>> rational = (1, 3)
>>> tree.quantize_to_rational(rational)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 3), {'name': 'a'}),
    TimeInterval(Offset(1, 3), Offset(2, 1), {'name': 'b'}),
    TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'c'}),
    TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'd'})
])

```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.scale_by_rational(rational)`

Scale aggregate duration of tree by *rational*:

```

>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> one = TimeInterval(0, 1, {'name': 'one'})
>>> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
>>> three = TimeInterval(2, 4, {'name': 'three'})
>>> tree = TimeIntervalTree([one, two, three])
>>> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

>>> result = tree.scale_by_rational((2, 3))
>>> result
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(2, 3), {'name': 'one'}),
    TimeInterval(Offset(1, 3), Offset(5, 3), {'name': 'two'}),
    TimeInterval(Offset(4, 3), Offset(8, 3), {'name': 'three'})
])

```

Scaling works regardless of the starting offset of the *TimeIntervalTree*:

```

>>> zero = TimeInterval(-4, 0, {'name': 'zero'})
>>> tree = TimeIntervalTree([zero, one, two, three])
>>> tree
TimeIntervalTree([
    TimeInterval(Offset(-4, 1), Offset(0, 1), {'name': 'zero'}),
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),

```

```

    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

>>> result = tree.scale_by_rational(2)
>>> result
TimeIntervalTree([
    TimeInterval(Offset(-4, 1), Offset(4, 1), {'name': 'zero'}),
    TimeInterval(Offset(4, 1), Offset(6, 1), {'name': 'one'}),
    TimeInterval(Offset(5, 1), Offset(9, 1), {'name': 'two'}),
    TimeInterval(Offset(8, 1), Offset(12, 1), {'name': 'three'})
])

>>> result.start == tree.start
True
>>> result.duration == tree.duration * 2
True

```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.scale_to_rational(rational)`

Scale aggregate duration of tree to *rational*:

```

>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> one = TimeInterval(0, 1, {'name': 'one'})
>>> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
>>> three = TimeInterval(2, 4, {'name': 'three'})
>>> tree = TimeIntervalTree([one, two, three])
>>> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

>>> result = tree.scale_to_rational(1)
>>> result
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 4), {'name': 'one'}),
    TimeInterval(Offset(1, 8), Offset(5, 8), {'name': 'two'}),
    TimeInterval(Offset(1, 2), Offset(1, 1), {'name': 'three'})
])

>>> result.scale_to_rational(10)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(5, 2), {'name': 'one'}),
    TimeInterval(Offset(5, 4), Offset(25, 4), {'name': 'two'}),
    TimeInterval(Offset(5, 1), Offset(10, 1), {'name': 'three'})
])

```

Scaling works regardless of the starting offset of the *TimeIntervalTree*:

```

>>> zero = TimeInterval(-4, 0, {'name': 'zero'})
>>> tree = TimeIntervalTree([zero, one, two, three])
>>> tree
TimeIntervalTree([
    TimeInterval(Offset(-4, 1), Offset(0, 1), {'name': 'zero'}),
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),

```

```

    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

>>> tree.scale_to_rational(4)
TimeIntervalTree([
    TimeInterval(Offset(-4, 1), Offset(-2, 1), {'name': 'zero'}),
    TimeInterval(Offset(-2, 1), Offset(-3, 2), {'name': 'one'}),
    TimeInterval(Offset(-7, 4), Offset(-3, 4), {'name': 'two'}),
    TimeInterval(Offset(-1, 1), Offset(0, 1), {'name': 'three'})
])

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.**shift_by_rational**(*rational*)

Shift aggregate offset of tree by *rational*:

```

>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> one = TimeInterval(0, 1, {'name': 'one'})
>>> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
>>> three = TimeInterval(2, 4, {'name': 'three'})
>>> tree = TimeIntervalTree([one, two, three])
>>> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

>>> result = tree.shift_by_rational(-2.5)
>>> result
TimeIntervalTree([
    TimeInterval(Offset(-5, 2), Offset(-3, 2), {'name': 'one'}),
    TimeInterval(Offset(-2, 1), Offset(0, 1), {'name': 'two'}),
    TimeInterval(Offset(-1, 2), Offset(3, 2), {'name': 'three'})
])

>>> result.shift_by_rational(6)
TimeIntervalTree([
    TimeInterval(Offset(7, 2), Offset(9, 2), {'name': 'one'}),
    TimeInterval(Offset(4, 1), Offset(6, 1), {'name': 'two'}),
    TimeInterval(Offset(11, 2), Offset(15, 2), {'name': 'three'})
])

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.**shift_to_rational**(*rational*)

Shift aggregate offset of tree to *rational*:

```

>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> one = TimeInterval(0, 1, {'name': 'one'})
>>> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
>>> three = TimeInterval(2, 4, {'name': 'three'})
>>> tree = TimeIntervalTree([one, two, three])
>>> tree
TimeIntervalTree([

```



```

    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

>>> result = tree.shift_to_rational(100)
>>> result
TimeIntervalTree([
    TimeInterval(Offset(100, 1), Offset(101, 1), {'name': 'one'}),
    TimeInterval(Offset(201, 2), Offset(205, 2), {'name': 'two'}),
    TimeInterval(Offset(102, 1), Offset(104, 1), {'name': 'three'})
])

```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.split_at_rationals(*rationals)`

Split tree at each rational in *rationals*:

```

>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> one = TimeInterval(0, 1, {'name': 'one'})
>>> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
>>> three = TimeInterval(2, 4, {'name': 'three'})
>>> tree = TimeIntervalTree([one, two, three])
>>> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

>>> result = tree.split_at_rationals(1, 2, 3)
>>> len(result)
4

>>> result[0]
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(1, 1), {'name': 'two'})
])

>>> result[1]
TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'two'})
])

>>> result[2]
TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'three'})
])

>>> result[3]
TimeIntervalTree([
    TimeInterval(Offset(3, 1), Offset(4, 1), {'name': 'three'})
])

```

Return tuple of *TimeIntervalTree* instances.

Special Methods

`TimeIntervalTree.__contains__(item)`

`TimeIntervalTree.__eq__(other)`

`TimeIntervalTree.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimeIntervalTree.__getitem__(item)`

`TimeIntervalTree.__getslice__(start, end)`

`TimeIntervalTree.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TimeIntervalTree.__iter__()`

`TimeIntervalTree.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimeIntervalTree.__len__()`

`TimeIntervalTree.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TimeIntervalTree.__ne__(other)`

`TimeIntervalTree.__nonzero__()`

TimeIntervalTree evaluates to True if it contains any intervals:

```
>>> from abjad.tools.timeintervaltools import *
>>> true_tree = TimeIntervalTree([TimeInterval(0, 1)])
>>> false_tree = TimeIntervalTree([])

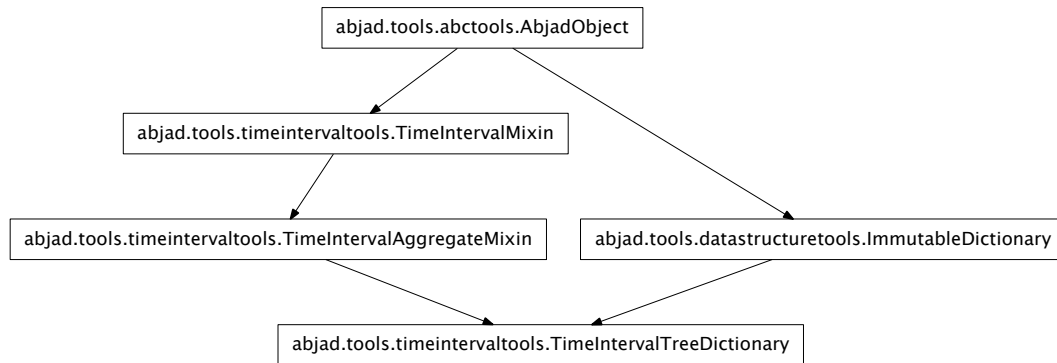
>>> bool(true_tree)
True
>>> bool(false_tree)
False
```

Return boolean.

`TimeIntervalTree.__repr__()`

`TimeIntervalTree.__str__()`

Inherited from `timeintervaltools._RedBlackTree`

timeintervaltools.TimeIntervalTreeDictionary

class `timeintervaltools.TimeIntervalTreeDictionary(*args)`

A dictionary of *TimeIntervalTrees*:

```

>>> from abjad.tools.timeintervaltools import TimeIntervalTreeDictionary

>>> from abjad.tools.timeintervaltools import TimeIntervalTree
>>> from abjad.tools.timeintervaltools import TimeInterval

>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})

```

TimeIntervalTreeDictionary can be instantiated from one or more other *TimeIntervalTreeDictionary* instances, whose trees will be fused if they share keys. It can also be instantiated from a regular dictionary whose values are *TimeIntervalTree* instances, or from a list of pairs where the second value of each pair is a *TimeIntervalTree* instance.

TimeIntervalTreeDictionary supports the same set of methods and properties as *TimeIntervalTree* and *TimeInterval*, including searching for intervals, quantizing, scaling, shifting and splitting.

TimeIntervalTreeDictionary is immutable.

Return *TimeIntervalTreeDictionary* instance.

Read-only Properties

TimeIntervalTreeDictionary.**bounds**

Start and stop of self returned as *TimeInterval* instance:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> bounds = interval.bounds
>>> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
>>> bounds == interval
True
>>> bounds is interval
False
```

Returns *TimeInterval* instance.

Inherited from *timeintervaltools.TimeIntervalMixin*

TimeIntervalTreeDictionary.**center**

Center offset of start and stop offsets:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.center
Offset(6, 1)
```

Returns *Offset* instance.

Inherited from *timeintervaltools.TimeIntervalMixin*

TimeIntervalTreeDictionary.**composite_tree**

The *TimeIntervalTree* composed of all the intervals in all trees in self:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

>>> treedict.composite_tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'})
])
```

Return *TimeIntervalTree* instance.

TimeIntervalTreeDictionary.**duration**

Duration of the time interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.duration
Duration(8, 1)
```

Returns *Duration* instance.

Inherited from *timeintervaltools.TimeIntervalMixin*

TimeIntervalTreeDictionary.earliest_start

The earliest start offset of all intervals in all trees in self:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

>>> treedict.earliest_start
Offset(0, 1)
```

Return *Offset* instance.

TimeIntervalTreeDictionary.earliest_stop

The earliest stop offset of all intervals in all trees in self:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

>>> treedict.earliest_stop
Offset(1, 1)
```

Return *Offset* instance.

TimeIntervalTreeDictionary.intervals**TimeIntervalTreeDictionary.latest_start**

The latest start offset of all intervals in all trees in self:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

>>> treedict.latest_start
Offset(2, 1)
```

Return *Offset* instance.

TimeIntervalTreeDictionary.latest_stop

The latest stop offset of all intervals in all trees in self:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

>>> treedict.latest_stop
Offset(3, 1)
```

Return *Offset* instance.

TimeIntervalTreeDictionary.offset_counts

Inherited from `timeintervaltools.TimeIntervalAggregateMixin`

`TimeIntervalTreeDictionary.offsets`

Inherited from `timeintervaltools.TimeIntervalAggregateMixin`

`TimeIntervalTreeDictionary.signature`

Tuple of start bound and stop bound.

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.signature
(Offset(2, 1), Offset(10, 1))
```

Returns 2-tuple of *Offset* instances.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.start`

Starting offset of interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.start
Offset(2, 1)
```

Returns *Offset* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.stop`

Stopping offset of interval:

```
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> interval = TimeInterval(2, 10)
>>> interval.stop
Offset(10, 1)
```

Returns *Offset* instance.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TimeIntervalTreeDictionary.clear()` → None. Remove all items from D.

Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.copy()` → a shallow copy of D

Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.find_intervals_intersecting_or_tangent_to_interval(*args)`

Find all intervals in dictionary intersecting or tangent to the interval defined in *args*:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
```

```

    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> interval = TimeInterval(0, 1)
>>> treedict.find_intervals_intersecting_or_tangent_to_interval(interval)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
})

>>> interval = TimeInterval(3, 4)
>>> treedict.find_intervals_intersecting_or_tangent_to_interval(interval)
TimeIntervalTreeDictionary({
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_intersecting_or_tangent_to_offset` (*offset*)

Find all intervals in dictionary intersecting or tangent to *offset*:

```

>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
})

```

```

    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> offset = 1
>>> treedict.find_intervals_intersecting_or_tangent_to_offset(offset)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
})

>>> offset = 3
>>> treedict.find_intervals_intersecting_or_tangent_to_offset(offset)
TimeIntervalTreeDictionary({
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_starting_after_offset` (*offset*)

Find all intervals in dictionary starting after *offset*:

```

>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> offset = 0
>>> treedict.find_intervals_starting_after_offset(offset)

```



```

TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> offset = 1
>>> treedict.find_intervals_starting_after_offset(offset)
TimeIntervalTreeDictionary({
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.find_intervals_starting_and_stopping_within_interval(*args)
Find all intervals in dictionary starting and stopping within the interval defined by *args*:

```

>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> interval = TimeInterval(1, 3)
>>> treedict.find_intervals_starting_and_stopping_within_interval(interval)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> interval = TimeInterval(-1, 2)
>>> treedict.find_intervals_starting_and_stopping_within_interval(interval)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
})

```

```
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**find_intervals_starting_at_offset** (*offset*)

Find all intervals in dictionary starting at *offset*:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> offset = 0
>>> treedict.find_intervals_starting_at_offset(offset)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
})

>>> offset = 1
>>> treedict.find_intervals_starting_at_offset(offset)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**find_intervals_starting_before_offset** (*offset*)

Find all intervals in dictionary starting before *offset*:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
```

```

>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})

>>> offset = 1
>>> treedict.find_intervals_starting_before_offset(offset)
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
})

>>> offset = 2
>>> treedict.find_intervals_starting_before_offset(offset)
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.find_intervals_starting_or_stopping_at_offset (*offset*)
Find all intervals in dictionary starting or stopping at *offset*:

```

>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([

```

```

        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> offset = 2
>>> treedict.find_intervals_starting_or_stopping_at_offset(offset)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> offset = 1
>>> treedict.find_intervals_starting_or_stopping_at_offset(offset)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_starting_within_interval(*args)`
 Find all intervals in dictionary starting within the interval defined by *args*:

```

>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

```

>>> interval = TimeInterval((-1, 2), (1, 2))
>>> treedict.find_intervals_starting_within_interval(interval)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
})

>>> interval = TimeInterval((1, 2), (5, 2))
>>> treedict.find_intervals_starting_within_interval(interval)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**find_intervals_stopping_after_offset** (*offset*)

Find all intervals in dictionary stopping after *offset*:

```

>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

>>> offset = 1
>>> treedict.find_intervals_stopping_after_offset(offset)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

```

    ]),
  })

>>> offset = 2
>>> treedict.find_intervals_stopping_after_offset(offset)
TimeIntervalTreeDictionary({
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**find_intervals_stopping_at_offset**(*offset*)

Find all intervals in dictionary stopping at *offset*:

```

>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})

>>> offset = 3
>>> treedict.find_intervals_stopping_at_offset(offset)
TimeIntervalTreeDictionary({
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})

>>> offset = 1
>>> treedict.find_intervals_stopping_at_offset(offset)
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_stopping_before_offset` (*offset*)

Find all intervals in dictionary stopping before *offset*:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})

>>> offset = 3
>>> treedict.find_intervals_stopping_before_offset(offset)
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
})

>>> offset = (7, 2)
>>> treedict.find_intervals_stopping_before_offset(offset)
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_stopping_within_interval` (**args*)

Find all intervals in dictionary stopping within the interval defined by *args*:

```
>>> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})

>>> interval = TimeInterval((3, 2), (5, 2))
>>> treedict.find_intervals_stopping_within_interval(interval)
TimeIntervalTreeDictionary({
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
})

>>> interval = TimeInterval((5, 2), (7, 2))
>>> treedict.find_intervals_stopping_within_interval(interval)
TimeIntervalTreeDictionary({
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**get**(*k*[, *d*]) → *D*[*k*] if *k* in *D*, else *d*. *d* defaults to None.

Inherited from `__builtin__.dict`

TimeIntervalTreeDictionary.**get_overlap_with_interval**(*interval*)

Return amount of overlap with *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTreeDictionary.**has_key**(*k*) → True if *D* has a key *k*, else False

Inherited from `__builtin__.dict`

TimeIntervalTreeDictionary.**is_contained_by_interval**(*interval*)

True if *interval* is contained by *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTreeDictionary.**is_container_of_interval**(*interval*)

True if *interval* contains *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.is_overlapped_by_interval(interval)`
True if `interval` is overlapped by *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.is_tangent_to_interval(interval)`
True if `interval` is tangent to *interval*.

Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.items()` → list of D's (key, value) pairs, as 2-tuples
Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.iteritems()` → an iterator over the (key, value) items of D
Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.iterkeys()` → an iterator over the keys of D
Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.itervalues()` → an iterator over the values of D
Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.keys()` → list of D's keys
Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised
Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.
Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.quantize_to_rational(rational)`
Quantize all intervals in dictionary to a multiple (1 or more) of *rational*:

```
>>> a = TimeIntervalTree([TimeInterval((1, 16), (1, 8), {'name': 'a'})])
>>> b = TimeIntervalTree([TimeInterval((2, 7), (13, 7), {'name': 'b'})])
>>> c = TimeIntervalTree([TimeInterval((3, 5), (8, 5), {'name': 'c'})])
>>> d = TimeIntervalTree([TimeInterval((2, 3), (5, 3), {'name': 'd'})])
>>> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
>>> treedict
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(1, 16), Offset(1, 8), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(2, 7), Offset(13, 7), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(3, 5), Offset(8, 5), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'd'}),
  ]),
})
```

```
>>> rational = (1, 4)
>>> treedict.quantize_to_rational(rational)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 4), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 4), Offset(7, 4), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(3, 2), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(3, 4), Offset(7, 4), {'name': 'd'}),
    ]),
})

>>> rational = (1, 3)
>>> treedict.quantize_to_rational(rational)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 3), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 3), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'd'}),
    ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.scale_by_rational(rational)`

Scale aggregate duration of dictionary by *rational*:

```
>>> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
>>> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
>>> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
>>> treedict = TimeIntervalTreeDictionary(
...     {'one': one, 'two': two, 'three': three})
>>> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
})

>>> result = treedict.scale_by_rational((2, 3))
>>> result
```

```
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(2, 3), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(4, 3), Offset(8, 3), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 3), Offset(5, 3), {'name': 'two'}),
    ]),
})
```

Scaling works regardless of the starting offset of the *TimeIntervalTreeDictionary*:

```
>>> zero = TimeIntervalTree([TimeInterval(-4, 0, {'name': 'zero'})])
>>> treedict = TimeIntervalTreeDictionary(
...     {'zero': zero, 'one': one, 'two': two, 'three': three})
>>> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
    'zero': TimeIntervalTree([
        TimeInterval(Offset(-4, 1), Offset(0, 1), {'name': 'zero'}),
    ]),
})

>>> result = treedict.scale_by_rational(2)
>>> result
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(4, 1), Offset(6, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(8, 1), Offset(12, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(5, 1), Offset(9, 1), {'name': 'two'}),
    ]),
    'zero': TimeIntervalTree([
        TimeInterval(Offset(-4, 1), Offset(4, 1), {'name': 'zero'}),
    ]),
})

>>> result.start == treedict.start
True
>>> result.duration == treedict.duration * 2
True
```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.scale_to_rational(rational)`
Scale aggregate duration of dictionary to *rational*:

```
>>> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
>>> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
>>> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
>>> treedict = TimeIntervalTreeDictionary({'one': one, 'two': two, 'three': three})
>>> treedict
TimeIntervalTreeDictionary({
  'one': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
  ]),
  'three': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
  ]),
  'two': TimeIntervalTree([
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
  ]),
})

>>> result = treedict.scale_to_rational(1)
>>> result
TimeIntervalTreeDictionary({
  'one': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 4), {'name': 'one'}),
  ]),
  'three': TimeIntervalTree([
    TimeInterval(Offset(1, 2), Offset(1, 1), {'name': 'three'}),
  ]),
  'two': TimeIntervalTree([
    TimeInterval(Offset(1, 8), Offset(5, 8), {'name': 'two'}),
  ]),
})

>>> result.scale_to_rational(10)
TimeIntervalTreeDictionary({
  'one': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(5, 2), {'name': 'one'}),
  ]),
  'three': TimeIntervalTree([
    TimeInterval(Offset(5, 1), Offset(10, 1), {'name': 'three'}),
  ]),
  'two': TimeIntervalTree([
    TimeInterval(Offset(5, 4), Offset(25, 4), {'name': 'two'}),
  ]),
})
```

Scaling works regardless of the starting offset of the *TimeIntervalTreeDictionary*:

```
>>> zero = TimeIntervalTree([TimeInterval(-4, 0, {'name': 'zero'})])
>>> treedict = TimeIntervalTreeDictionary(
...     {'zero': zero, 'one': one, 'two': two, 'three': three})

>>> treedict.scale_to_rational(4)
TimeIntervalTreeDictionary({
  'one': TimeIntervalTree([
    TimeInterval(Offset(-2, 1), Offset(-3, 2), {'name': 'one'}),
  ]),
  'three': TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(0, 1), {'name': 'three'}),
  ]),
  'two': TimeIntervalTree([
```

```

        TimeInterval(Offset(-7, 4), Offset(-3, 4), {'name': 'two'}),
    ]),
    'zero': TimeIntervalTree([
        TimeInterval(Offset(-4, 1), Offset(-2, 1), {'name': 'zero'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**setdefault** (*k*[, *d*]) → *D.get(k,d)*, also set *D[k]=d* if *k* not in *D*
 Inherited from `__builtin__.dict`

TimeIntervalTreeDictionary.**shift_by_rational** (*rational*)

Shift aggregate offset of dictionary by *rational*:

```

>>> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
>>> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
>>> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
>>> treedict = TimeIntervalTreeDictionary({'one': one, 'two': two, 'three': three})
>>> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
})

>>> result = treedict.shift_by_rational(-2.5)
>>> result
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(-5, 2), Offset(-3, 2), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(-1, 2), Offset(3, 2), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(-2, 1), Offset(0, 1), {'name': 'two'}),
    ]),
})

>>> result.shift_by_rational(6)
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(7, 2), Offset(9, 2), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(11, 2), Offset(15, 2), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(4, 1), Offset(6, 1), {'name': 'two'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.shift_to_rational(rational)`

Shift aggregate offset of dictionary to *rational*:

```
>>> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
>>> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
>>> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
>>> treedict = TimeIntervalTreeDictionary({'one': one, 'two': two, 'three': three})
>>> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
})

>>> result = treedict.shift_to_rational(100)
>>> result
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(100, 1), Offset(101, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(102, 1), Offset(104, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(201, 2), Offset(205, 2), {'name': 'two'}),
    ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.split_at_rationals(*rationals)`

Split dictionary at each rational in *rationals*:

```
>>> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
>>> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
>>> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
>>> treedict = TimeIntervalTreeDictionary({'one': one, 'two': two, 'three': three})
>>> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
})
```

```
>>> result = treedict.split_at_rationals(1, 2, 3)
>>> len(result)
4

>>> result[0]
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(1, 1), {'name': 'two'}),
    ]),
})

>>> result[1]
TimeIntervalTreeDictionary({
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'two'}),
    ]),
})

>>> result[2]
TimeIntervalTreeDictionary({
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(5, 2), {'name': 'two'}),
    ]),
})

>>> result[3]
TimeIntervalTreeDictionary({
    'three': TimeIntervalTree([
        TimeInterval(Offset(3, 1), Offset(4, 1), {'name': 'three'}),
    ]),
})
```

Return tuple of *TimeIntervalTreeDictionary* instances.

TimeIntervalTreeDictionary.**update** (*[E]*, ***F*) → None. Update D from dict/iterable E and F.
 If E present and has a *.keys()* method, does: for k in E: D[k] = E[k] If E present and lacks *.keys()* method, does:
 for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Inherited from *__builtin__.dict*

TimeIntervalTreeDictionary.**values** () → list of D's values

Inherited from *__builtin__.dict*

TimeIntervalTreeDictionary.**viewitems** () → a set-like object providing a view on D's items

Inherited from *__builtin__.dict*

TimeIntervalTreeDictionary.**viewkeys** () → a set-like object providing a view on D's keys

Inherited from *__builtin__.dict*

TimeIntervalTreeDictionary.**viewvalues** () → an object providing a view on D's values

Inherited from *__builtin__.dict*

Special Methods

```

TimeIntervalTreeDictionary.__cmp__(y) <==> cmp(x, y)
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__contains__(k) → True if D has a key k, else False
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__delitem__(*args)
    Inherited from datastructuretools.ImmutableDictionary
TimeIntervalTreeDictionary.__eq__()
    x.__eq__(y) <==> x==y
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__getitem__()
    x.__getitem__(y) <==> x[y]
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__iter__() <==> iter(x)
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__len__() <==> len(x)
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.dict
TimeIntervalTreeDictionary.__nonzero__()
    Inherited from timeintervaltools.TimeIntervalMixin
TimeIntervalTreeDictionary.__repr__()
TimeIntervalTreeDictionary.__setitem__(*args)
    Inherited from datastructuretools.ImmutableDictionary

```

functions

timeintervaltools.all_are_intervals_or_trees_or_empty

```

timeintervaltools.all_are_intervals_or_trees_or_empty(input)
    Recursively test if all elements of input are TimeIntervals or TimeIntervalTrees. An empty result also return as

```


True.

timeintervaltools.all_intervals_are_contiguous

`timeintervaltools.all_intervals_are_contiguous(intervals)`

True when all intervals in *intervals* are contiguous and non-overlapping.

timeintervaltools.all_intervals_are_nonoverlapping

`timeintervaltools.all_intervals_are_nonoverlapping(intervals)`

True when all intervals in *intervals* in tree are non-overlapping.

timeintervaltools.calculate_density_of_attacks_in_interval

`timeintervaltools.calculate_density_of_attacks_in_interval(intervals, interval)`

Calculate the number of attacks in *interval* over the duration of *interval*.

Return Fraction.

timeintervaltools.calculate_density_of_releases_in_interval

`timeintervaltools.calculate_density_of_releases_in_interval(intervals, interval)`

Calculate the number of releases in *interval* divided by the duration of *interval*.

Return Fraction.

timeintervaltools.calculate_depth_centroid_of_intervals

`timeintervaltools.calculate_depth_centroid_of_intervals(intervals)`

Calculate the weighted mean offset of *intervals*, such that the centroids of each interval in the depth tree of *intervals* make up the values of the mean, and the depth of each interval in the depth tree of *intervals* make up the weights.

Return Offset.

timeintervaltools.calculate_depth_centroid_of_intervals_in_interval

`timeintervaltools.calculate_depth_centroid_of_intervals_in_interval(intervals, interval)`

Return the weighted mean of the depth tree of *intervals* in *interval*, such that the centroids of each interval of the depth tree are the values, and the weights are the depths at each interval of the depth tree.

timeintervaltools.calculate_depth_density_of_intervals

`timeintervaltools.calculate_depth_density_of_intervals(intervals)`

Return a Fraction, of the duration of each interval in the depth tree of *intervals*, multiplied by the depth at that interval, divided by the overall duration of *intervals*.

The depth density of a single interval is 1

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(0, 1)
>>> b = TimeInterval(0, 1)
>>> c = TimeInterval(Fraction(1, 2), 1)
>>> timeintervaltools.calculate_depth_density_of_intervals(a)
Duration(1, 1)
>>> timeintervaltools.calculate_depth_density_of_intervals([a, b])
Duration(2, 1)
>>> timeintervaltools.calculate_depth_density_of_intervals([a, c])
Duration(3, 2)
>>> timeintervaltools.calculate_depth_density_of_intervals([a, b, c])
Duration(5, 2)
```

Return fraction.

timeintervaltools.calculate_depth_density_of_intervals_in_interval

timeintervaltools.calculate_depth_density_of_intervals_in_interval (*intervals*,
interval)

Return a Fraction, of the duration of each interval in the depth tree of *intervals* within *interval*, multiplied by the depth at that interval, divided by the overall duration of *intervals*.

timeintervaltools.calculate_mean_attack_of_intervals

timeintervaltools.calculate_mean_attack_of_intervals (*intervals*)

Return Fraction of the average attack offset of *intervals*

timeintervaltools.calculate_mean_release_of_intervals

timeintervaltools.calculate_mean_release_of_intervals (*intervals*)

Return a Fraction of the average release offset of *intervals*.

timeintervaltools.calculate_min_mean_and_max_depth_of_intervals

timeintervaltools.calculate_min_mean_and_max_depth_of_intervals (*intervals*)

Return a 3-tuple of the minimum, mean and maximum depth of *intervals*. If *intervals* is empty, return None. “Mean” in this case is a weighted mean, where the durations of the intervals in depth tree of *intervals* are the weights

timeintervaltools.calculate_min_mean_and_max_durations_of_intervals

timeintervaltools.calculate_min_mean_and_max_durations_of_intervals (*intervals*)

Return a 3-tuple of the minimum, mean and maximum duration of all intervals in *intervals*. If *intervals* is empty, return None.

timeintervaltools.calculate_sustain_centroid_of_intervals

`timeintervaltools.calculate_sustain_centroid_of_intervals` (*intervals*)

Return a weighted mean, such that the centroid of each interval in *intervals* are the values, and the weights are their durations.

timeintervaltools.clip_interval_durations_to_range

`timeintervaltools.clip_interval_durations_to_range` (*intervals*, *minimum=None*, *maximum=None*)

timeintervaltools.compute_depth_of_intervals

`timeintervaltools.compute_depth_of_intervals` (*intervals*)

Compute a tree whose intervals represent the depth (level of overlap) in each boundary pair of *intervals*:

```
>>> from abjad.tools.timeintervaltools import *
>>> a = TimeInterval(0, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 15)
>>> tree = TimeIntervalTree([a, b, c])
>>> compute_depth_of_intervals(tree)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'depth': 1}),
    TimeInterval(Offset(3, 1), Offset(6, 1), {'depth': 0}),
    TimeInterval(Offset(6, 1), Offset(9, 1), {'depth': 1}),
    TimeInterval(Offset(9, 1), Offset(12, 1), {'depth': 2}),
    TimeInterval(Offset(12, 1), Offset(15, 1), {'depth': 1})
])
```

Return interval tree.

timeintervaltools.compute_depth_of_intervals_in_interval

`timeintervaltools.compute_depth_of_intervals_in_interval` (*intervals*, *interval*)

Compute a tree whose intervals represent the depth (level of overlap) in each boundary pair of *intervals*, cropped within *interval*:

```
>>> from abjad.tools.timeintervaltools import *
>>> a = TimeInterval(0, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 15)
>>> tree = TimeIntervalTree([a, b, c])
>>> d = TimeInterval(-1, 16)
>>> compute_depth_of_intervals_in_interval(tree, d)
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(0, 1), {'depth': 0}),
    TimeInterval(Offset(0, 1), Offset(3, 1), {'depth': 1}),
    TimeInterval(Offset(3, 1), Offset(6, 1), {'depth': 0}),
    TimeInterval(Offset(6, 1), Offset(9, 1), {'depth': 1}),
    TimeInterval(Offset(9, 1), Offset(12, 1), {'depth': 2}),
    TimeInterval(Offset(12, 1), Offset(15, 1), {'depth': 1}),
    TimeInterval(Offset(15, 1), Offset(16, 1), {'depth': 0})
])
```

Return interval tree.

timeintervaltools.compute_logical_and_of_intervals

`timeintervaltools.compute_logical_and_of_intervals(intervals)`

Compute the logical AND of a collection of intervals.

Return TimeIntervalTree.

timeintervaltools.compute_logical_and_of_intervals_in_interval

`timeintervaltools.compute_logical_and_of_intervals_in_interval(intervals, interval)`

Compute the logical AND of a collection of intervals, cropped within *interval*.

Return TimeIntervalTree.

timeintervaltools.compute_logical_not_of_intervals

`timeintervaltools.compute_logical_not_of_intervals(intervals)`

Compute the logical NOT of some collection of intervals.

Return TimeIntervalTree.

timeintervaltools.compute_logical_not_of_intervals_in_interval

`timeintervaltools.compute_logical_not_of_intervals_in_interval(intervals, interval)`

Compute the logical NOT of some collection of intervals, cropped within *interval*.

Return TimeIntervalTree.

timeintervaltools.compute_logical_or_of_intervals

`timeintervaltools.compute_logical_or_of_intervals(intervals)`

Compute the logical OR of a collection of intervals.

Return TimeIntervalTree.

timeintervaltools.compute_logical_or_of_intervals_in_interval

`timeintervaltools.compute_logical_or_of_intervals_in_interval(intervals, interval)`

Compute the logical OR of a collection of intervals, cropped within *interval*.

Return TimeIntervalTree.

timeintervaltools.compute_logical_xor_of_intervals

`timeintervaltools.compute_logical_xor_of_intervals` (*intervals*)

Compute the logical XOR of a collections of intervals.

Return `TimeIntervalTree`.

timeintervaltools.compute_logical_xor_of_intervals_in_interval

`timeintervaltools.compute_logical_xor_of_intervals_in_interval` (*intervals*, *interval*)

Compute the logical XOR of a collections of intervals, cropped within *interval*.

Return `TimeIntervalTree`.

timeintervaltools.concatenate_trees

`timeintervaltools.concatenate_trees` (*trees*, *padding=0*)

Merge all trees in *trees*, offsetting each subsequent tree to start after the previous.

Return `TimeIntervalTree`.

timeintervaltools.explode_intervals_compactly

`timeintervaltools.explode_intervals_compactly` (*intervals*)

Explode the intervals in *intervals* into *n* non-overlapping trees, where *n* is the maximum depth of *intervals*.

The algorithm will attempt to insert the exploded intervals into the lowest-indexed resultant tree with free space.

Return an array of `TimeIntervalTree` instances.

timeintervaltools.explode_intervals_into_n_trees_heuristically

`timeintervaltools.explode_intervals_into_n_trees_heuristically` (*intervals*, *n*)

Explode *intervals* into *n* trees, avoiding overlap when possible, and distributing intervals so as to equalize density across the trees.

Return list of `TimeIntervalTree` instances.

timeintervaltools.explode_intervals_uncompactly

`timeintervaltools.explode_intervals_uncompactly` (*intervals*)

Explode the intervals in *intervals* into *n* non-overlapping trees, where *n* is the maximum depth of *intervals*.

The algorithm will attempt to insert the exploded intervals cyclically, making its insertion attempt at the next resultant tree in the array, rather than always beginning its search from index 0.

Return list of `TimeIntervalTree` instances.

timeintervaltools.fuse_overlapping_intervals

`timeintervaltools.fuse_overlapping_intervals` (*intervals*)

Fuse the overlapping intervals in *intervals* and return an *TimeIntervalTree* of the result

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(0, 10)
>>> b = TimeInterval(5, 15)
>>> c = TimeInterval(15, 25)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.fuse_overlapping_intervals(tree)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(15, 1), {}),
    TimeInterval(Offset(15, 1), Offset(25, 1), {})
])
```

Return *TimeIntervalTree*.

timeintervaltools.fuse_tangent_or_overlapping_intervals

`timeintervaltools.fuse_tangent_or_overlapping_intervals` (*intervals*)

Fuse all tangent or overlapping intervals and return an *TimeIntervalTree* of the result

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(0, 10)
>>> b = TimeInterval(5, 15)
>>> c = TimeInterval(15, 25)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.fuse_tangent_or_overlapping_intervals(tree)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(25, 1), {})
])
```

Return *TimeIntervalTree*.

timeintervaltools.get_all_unique_bounds_in_intervals

`timeintervaltools.get_all_unique_bounds_in_intervals` (*intervals*)

Find all unique starting and ending boundaries in *intervals*.

Return list of Offsets.

timeintervaltools.group_overlapping_intervals_and_yield_groups

`timeintervaltools.group_overlapping_intervals_and_yield_groups` (*intervals*)

Group overlapping intervals in *intervals*.

Yield *TimeIntervalTrees*.

timeintervaltools.group_tangent_or_overlapping_intervals_and_yield_groups

timeintervaltools.**group_tangent_or_overlapping_intervals_and_yield_groups** (*intervals*)

Group tangent or overlapping intervals in *intervals*.

Yield TimeIntervalTrees.

timeintervaltools.make_monophonic_percussion_score_from_nonoverlapping_intervals

timeintervaltools.**make_monophonic_percussion_score_from_nonoverlapping_intervals** (*intervals*,
col-
orkey=None)

Create a monophonic percussion score from nonoverlapping interval collection *intervals*.

Return Score.

timeintervaltools.make_polyphonic_percussion_score_from_nonoverlapping_trees

timeintervaltools.**make_polyphonic_percussion_score_from_nonoverlapping_trees** (*trees*,
col-
orkey=None)

Make a polyphonic percussion score from a collections of non-overlapping trees.

Return LilyPondFile.

timeintervaltools.make_voice_from_nonoverlapping_intervals

timeintervaltools.**make_voice_from_nonoverlapping_intervals** (*intervals*, col-
orkey=None,
pitch=None)

timeintervaltools.mask_intervals_with_intervals

timeintervaltools.**mask_intervals_with_intervals** (*masked_intervals*, *mask_intervals*)
Clip or remove all intervals in *masked_intervals* outside of the bounds defined in *mask_intervals*, while main-
taining *masked_intervals*' payload contents

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(0, 10, {'a': 1})
>>> b = TimeInterval(5, 15, {'b': 2})
>>> tree = TimeIntervalTree([a, b])
>>> mask = TimeInterval(4, 11)
>>> timeintervaltools.mask_intervals_with_intervals(tree, mask)
TimeIntervalTree([
    TimeInterval(Offset(4, 1), Offset(10, 1), {'a': 1}),
    TimeInterval(Offset(5, 1), Offset(11, 1), {'b': 2})
])
```

Return TimeIntervalTree.

timeintervaltools.resolve_overlaps_between_nonoverlapping_trees

`timeintervaltools.resolve_overlaps_between_nonoverlapping_trees`(*trees*, *minimum_duration=None*)

Create a nonoverlapping `TimeIntervalTree` from *trees*. Intervals in higher-indexed trees in *trees* only appear in part or whole where they do not overlap intervals from starter-indexed trees

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeIntervalTree(TimeInterval(0, 4, {'a': 1}))
>>> b = TimeIntervalTree(TimeInterval(1, 5, {'b': 2}))
>>> c = TimeIntervalTree(TimeInterval(2, 6, {'c': 3}))
>>> d = TimeIntervalTree(TimeInterval(1, 3, {'d': 4}))
>>> timeintervaltools.resolve_overlaps_between_nonoverlapping_trees([a, b, c, d])
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(4, 1), {'a': 1}),
    TimeInterval(Offset(4, 1), Offset(5, 1), {'b': 2}),
    TimeInterval(Offset(5, 1), Offset(6, 1), {'c': 3})
])
```

Return `TimeIntervalTree`.

timeintervaltools.round_interval_bounds_to_nearest_multiple_of_rational

`timeintervaltools.round_interval_bounds_to_nearest_multiple_of_rational`(*intervals*, *duration*)

Round all start and stop offsets of *intervals* to the nearest multiple of *duration*.

If both start and stop of an interval collapse on the same offset, that interval's stop will be adjusted to the next larger multiple of *duration*.

Return `TimeIntervalTree`.

timeintervaltools.scale_aggregate_duration_by_rational

`timeintervaltools.scale_aggregate_duration_by_rational`(*intervals*, *rational*)

Scale the aggregate duration of all intervals in *intervals* by *rational*, maintaining the original start offset

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(-1, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 16)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.scale_aggregate_duration_by_rational(tree, Fraction(1, 3))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(1, 3), {}),
    TimeInterval(Offset(4, 3), Offset(10, 3), {}),
    TimeInterval(Offset(7, 3), Offset(14, 3), {})
])
```


Return `TimeIntervalTree`.

`timeintervaltools.scale_aggregate_duration_to_rational`

`timeintervaltools.scale_aggregate_duration_to_rational(intervals, rational)`

Scale the aggregate duration of all intervals in *intervals* to *rational*, maintaining the original start offset

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(-1, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 16)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.scale_aggregate_duration_to_rational(tree, Fraction(16, 7))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(-55, 119), {}),
    TimeInterval(Offset(-1, 17), Offset(89, 119), {}),
    TimeInterval(Offset(41, 119), Offset(9, 7), {})
])
```

Return `TimeIntervalTree`.

`timeintervaltools.scale_interval_durations_by_rational`

`timeintervaltools.scale_interval_durations_by_rational(intervals, rational)`

Scale the duration of each interval in *intervals* by *rational*, maintaining their start offsets

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(-1, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 16)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.scale_interval_durations_by_rational(tree, Fraction(6, 5))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(19, 5), {}),
    TimeInterval(Offset(6, 1), Offset(66, 5), {}),
    TimeInterval(Offset(9, 1), Offset(87, 5), {})
])
```

Return `TimeIntervalTree`.

`timeintervaltools.scale_interval_durations_to_rational`

`timeintervaltools.scale_interval_durations_to_rational(intervals, rational)`

Scale the duration of each interval in *intervals* to *rational*, maintaining their start offsets

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree
```

```
>>> a = TimeInterval(-1, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 16)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.scale_interval_durations_to_rational(tree, Fraction(1, 7))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(-6, 7), {}),
    TimeInterval(Offset(6, 1), Offset(43, 7), {}),
    TimeInterval(Offset(9, 1), Offset(64, 7), {})
])
```

Return `TimeIntervalTree`.

`timeintervaltools.scale_interval_offsets_by_rational`

`timeintervaltools.scale_interval_offsets_by_rational(intervals, rational)`

Scale the starting offset of each interval in *intervals* by *rational*, maintaining the startest offset in *intervals*

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(-1, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 16)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.scale_interval_offsets_by_rational(tree, Fraction(4, 5))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(3, 1), {}),
    TimeInterval(Offset(23, 5), Offset(53, 5), {}),
    TimeInterval(Offset(7, 1), Offset(14, 1), {})
])
```

Return interval tree.

`timeintervaltools.shift_aggregate_offset_by_rational`

`timeintervaltools.shift_aggregate_offset_by_rational(intervals, rational)`

Shift the aggregate offset of *intervals* by *rational*

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(-1, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 16)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.shift_aggregate_offset_by_rational(tree, Fraction(1, 3))
TimeIntervalTree([
    TimeInterval(Offset(-2, 3), Offset(10, 3), {}),
    TimeInterval(Offset(19, 3), Offset(37, 3), {}),
    TimeInterval(Offset(28, 3), Offset(49, 3), {})
])
```

Return `TimeIntervalTree`.

timeintervaltools.shift_aggregate_offset_to_rational

`timeintervaltools.shift_aggregate_offset_to_rational(intervals, rational)`

Shift the aggregate offset of *intervals* to *rational*

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(-1, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 16)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.shift_aggregate_offset_to_rational(tree, Fraction(10, 7))
TimeIntervalTree([
    TimeInterval(Offset(10, 7), Offset(38, 7), {}),
    TimeInterval(Offset(59, 7), Offset(101, 7), {}),
    TimeInterval(Offset(80, 7), Offset(129, 7), {})
])
```

Return `TimeIntervalTree`.

timeintervaltools.split_intervals_at_rationals

`timeintervaltools.split_intervals_at_rationals(intervals, offsets)`

Split *intervals* at each offset in *offsets*

```
>>> from abjad.tools import timeintervaltools
>>> from abjad.tools.timeintervaltools import TimeInterval
>>> from abjad.tools.timeintervaltools import TimeIntervalTree

>>> a = TimeInterval(-1, 3)
>>> b = TimeInterval(6, 12)
>>> c = TimeInterval(9, 16)
>>> tree = TimeIntervalTree([a, b, c])
>>> timeintervaltools.split_intervals_at_rationals(tree, [1, Fraction(19, 2)])
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(1, 1), {}),
    TimeInterval(Offset(1, 1), Offset(3, 1), {}),
    TimeInterval(Offset(6, 1), Offset(19, 2), {}),
    TimeInterval(Offset(9, 1), Offset(19, 2), {}),
    TimeInterval(Offset(19, 2), Offset(12, 1), {}),
    TimeInterval(Offset(19, 2), Offset(16, 1), {})
])
```

Return `TimeIntervalTree`.

timesignaturetools

functions

timesignaturetools.duration_and_possible_denominators_to_time_signature

```
timesignaturetools.duration_and_possible_denominators_to_time_signature(duration,
                                                                    de-
                                                                    nom-
                                                                    i-
                                                                    na-
                                                                    tors=None,
                                                                    fac-
                                                                    tor=None)
```

Make new meter equal to *duration*:

```
>>> from abjad.tools import timesignaturetools

>>> timesignaturetools.duration_and_possible_denominators_to_time_signature(Duration(3, 2))
TimeSignatureMark((3, 2))
```

Make new meter equal to *duration* with denominator equal to the first possible element in *denominators*:

```
>>> timesignaturetools.duration_and_possible_denominators_to_time_signature(
... Duration(3, 2), denominators=[5, 6, 7, 8])
TimeSignatureMark((9, 6))
```

Make new meter equal to *duration* with denominator divisible by *factor*:

```
>>> timesignaturetools.duration_and_possible_denominators_to_time_signature(
... Duration(3, 2), factor=5)
TimeSignatureMark((15, 10))
```

Return new meter. Changed in version 2.0: renamed `timesignaturetools.make_best()` to `timesignaturetools.duration_and_possible_denominators_to_time_signature()`.

timesignaturetools.get_nonbinary_factor_from_time_signature_denominator

```
timesignaturetools.get_nonbinary_factor_from_time_signature_denominator(time_signature)
```

Get nonbinary factor from nonbinary *time_signature* denominator:

```
>>> from abjad.tools import timesignaturetools

>>> timesignaturetools.get_nonbinary_factor_from_time_signature_denominator(
... contexttools.TimeSignatureMark((3, 12)))
3

>>> timesignaturetools.get_nonbinary_factor_from_time_signature_denominator(
... contexttools.TimeSignatureMark((3, 13)))
13

>>> timesignaturetools.get_nonbinary_factor_from_time_signature_denominator(
... contexttools.TimeSignatureMark((3, 14)))
7
```

Get 1 from binary *time_signature* denominator:

Return nonnegative integer.

`timesignaturetools.is_time_signature_with_equivalent_binary_representation(expr)`
True when *expr* is a meter with binary-valued duration:

Otherwise false:

Return boolean.

```
timesignaturetools.time_signature_to_binary_time_signature(nonbinary_meter,
                                                         contents_multiplier=Fraction(1,
                                                         1))
```

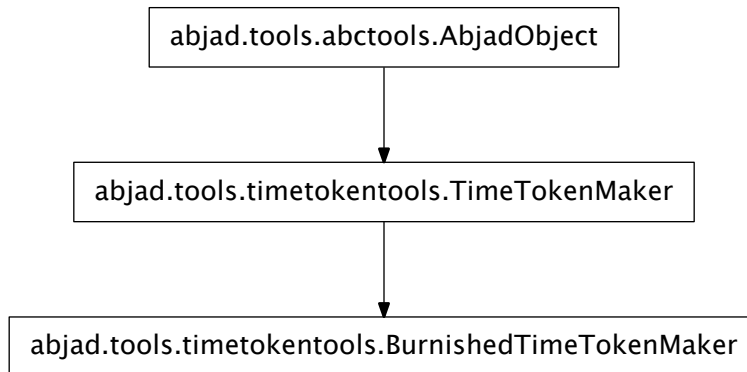
Preserve binary *meter*:

Return	newly constructed meter.	Changed in version 2.0:	renamed
timesignaturetools.make_binary()	to	timesignaturetools.time_signature_to_binary_time_sig	

`timetokentools`

abstract classes

`timetokentools.BurnishedTimeTokenMaker`



```

class timetokentools.BurnishedTimeTokenMaker (pattern,          denominator,          prola-
tion_addenda=None,          lefts=None,
middles=None,          rights=None,
left_lengths=None,          right_lengths=None,
secondary_divisions=None,          pat-
tern_helper=None,          prola-
tion_addenda_helper=None,
lefts_helper=None,          mid-
dles_helper=None,          rights_helper=None,
left_lengths_helper=None,
right_lengths_helper=None,          sec-
ondary_divisions_helper=None)
  
```

New in version 2.8. Abstract base class for time token makers that burnish some or all of the time tokens they produce.

‘Burnishing’ means to forcibly cast the first or last (or both first and last) elements of a time token to be either a note or rest.

‘Token-burnishing’ time token makers burnish every time token they produce.

‘Output-burnishing’ time token makers burnish only the first and last time tokens they produce and leave interior time tokens unchanged.

Read-only Properties

`BurnishedTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`BurnishedTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetoken.tools.TimeTokenMaker`

Special Methods

`BurnishedTimeTokenMaker.__call__(duration_tokens, seeds=None)`

`BurnishedTimeTokenMaker.__eq__(other)`

`BurnishedTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BurnishedTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BurnishedTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BurnishedTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BurnishedTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

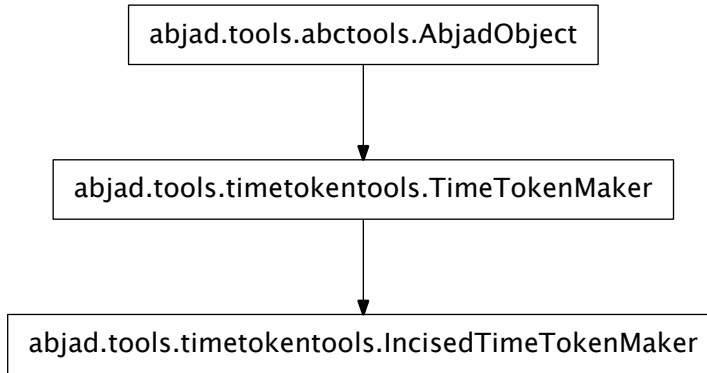
Return boolean.

Inherited from `abctools.AbjadObject`

`BurnishedTimeTokenMaker.__repr__()`

Inherited from `timetoken.tools.TimeTokenMaker`

timetokentools.IncisedTimeTokenMaker



```

class timetokentools.IncisedTimeTokenMaker (prefix_signal,    prefix_lengths,    suffix_signal,
                                             suffix_lengths,    denominator,    prola-
tion_addenda=None, secondary_divisions=None,
prefix_signal_helper=None,                                pre-
fix_lengths_helper=None,                                suf-
fix_signal_helper=None,                                suf-
fix_lengths_helper=None,                                prola-
tion_addenda_helper=None,                                sec-
ondary_divisions_helper=None)
  
```

Abstract base class for time token makers that incise some or all of the time tokens they produce.

Time token makers can incise the edge of every time token.

Or time token makers can incise only the start of the first time token and the end of the last time token.

Read-only Properties

`IncisedTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`IncisedTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```

self.pattern
self.prolation_addenda
  
```



```
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`IncisedTimeTokenMaker.__call__(duration_tokens, seeds=None)`

`IncisedTimeTokenMaker.__eq__(other)`

`IncisedTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

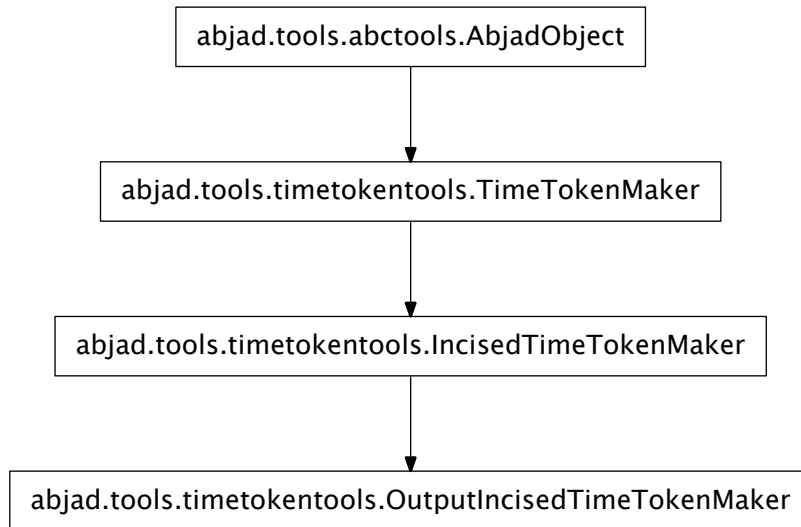
Return boolean.

Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__repr__()`

Inherited from `timetokentools.TimeTokenMaker`

timetokentools.OutputIncisedTimeTokenMaker



```

class timetokentools.OutputIncisedTimeTokenMaker (prefix_signal, prefix_lengths, suf-
fix_signal, suffix_lengths, denom-
inator, prolation_addenda=None,
secondary_divisions=None, pre-
fix_signal_helper=None, pre-
fix_lengths_helper=None, suf-
fix_signal_helper=None, suf-
fix_lengths_helper=None, prola-
tion_addenda_helper=None, sec-
ondary_divisions_helper=None)

```

New in version 2.8. Abstract base class for time token makers that incise only the first and last time tokens they produce.

Read-only Properties

`OutputIncisedTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`OutputIncisedTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`OutputIncisedTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedTimeTokenMaker.__eq__(other)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

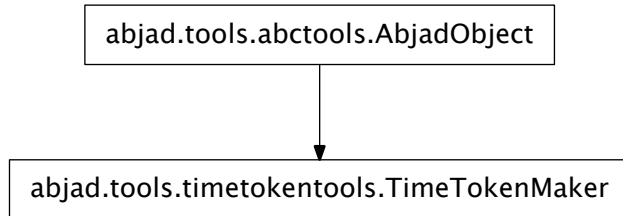
Return boolean.

Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__repr__()`

Inherited from `timetokentools.TimeTokenMaker`

timetokentools.TimeTokenMaker



class `timetokentools.TimeTokenMaker`

New in version 2.8. Time token maker abstract base class.

Read-only Properties

`TimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```

self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
  
```

Operate in place and return none.

Special Methods

`TimeTokenMaker.__call__(duration_tokens, seeds=None)`

`TimeTokenMaker.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TimeTokenMaker.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`TimeTokenMaker.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

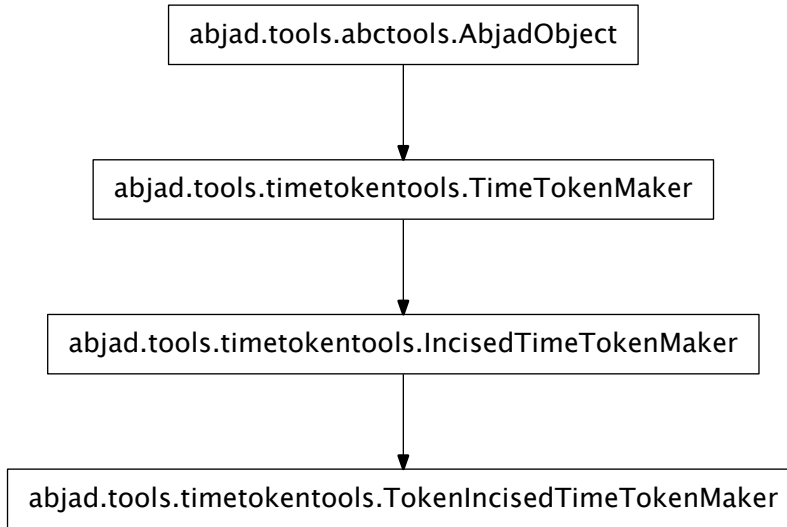
`TimeTokenMaker.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`TimeTokenMaker.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`TimeTokenMaker.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`TimeTokenMaker.__repr__()`

timetokentools.TokenIncisedTimeTokenMaker



```

class timetokentools.TokenIncisedTimeTokenMaker (prefix_signal,    prefix_lengths,    suf-
fix_signal,    suffix_lengths,    denom-
inator,    prolotion_addenda=None,
secondary_divisions=None,    pre-
fix_signal_helper=None,    pre-
fix_lengths_helper=None,    suf-
fix_signal_helper=None,    suf-
fix_lengths_helper=None,    prola-
tion_addenda_helper=None,    sec-
ondary_divisions_helper=None)

```

New in version 2.8. Abstract base class for time token makers that incise every time token they produce.

Read-only Properties

`TokenIncisedTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TokenIncisedTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`TokenIncisedTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`TokenIncisedTimeTokenMaker.__eq__(other)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`TokenIncisedTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

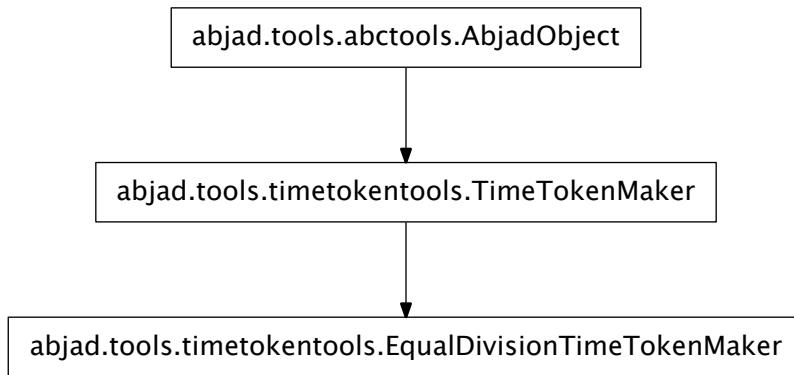
Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__repr__()`

Inherited from `timetokentools.TimeTokenMaker`

concrete classes

timetokentools.EqualDivisionTimeTokenMaker



class timetokentools.**EqualDivisionTimeTokenMaker** (*leaf_count*, *is_diminution=True*)

New in version 2.10. Equal division time-token maker:

```

>>> maker = timetokentools.EqualDivisionTimeTokenMaker(4)

>>> duration_tokens = [(1, 4), (1, 8), (1, 6), (1, 12)]
>>> tuplet_lists = maker(duration_tokens)
>>> tuplets = sequencetools.flatten_sequence(tuplet_lists)

>>> staff = Staff(tuplets)

>>> f(staff)
\new Staff {
  {
    c'16
    c'16
    c'16
    c'16
  }
  {
    c'32
    c'32
    c'32
    c'32
  }
  \times 2/3 {
    c'16
    c'16
    c'16
    c'16
  }
  \times 2/3 {
    c'32
  }
}
  
```



```
        c' 32
        c' 32
        c' 32
    }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`EqualDivisionTimeTokenMaker.is_diminution`

Return boolean.

`EqualDivisionTimeTokenMaker.leaf_count`

Leaf count.

Note: add example.

Return positive integer.

`EqualDivisionTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`EqualDivisionTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`EqualDivisionTimeTokenMaker.__call__(duration_tokens, seeds=None)`

`EqualDivisionTimeTokenMaker.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`EqualDivisionTimeTokenMaker.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

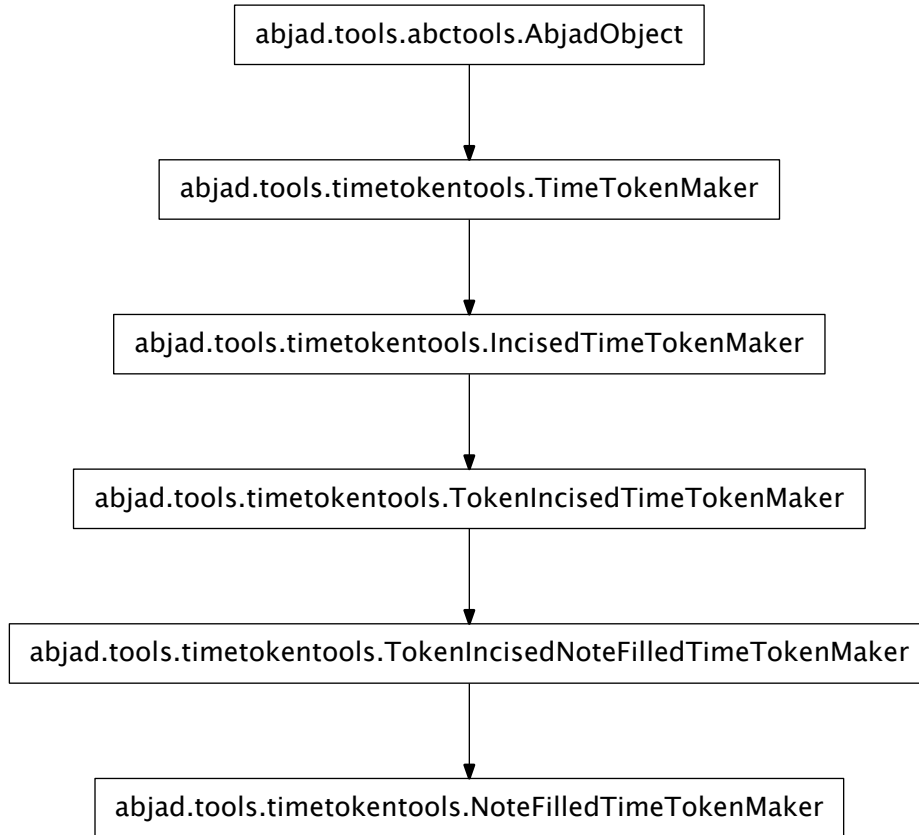
`EqualDivisionTimeTokenMaker.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`EqualDivisionTimeTokenMaker.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`EqualDivisionTimeTokenMaker.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`EqualDivisionTimeTokenMaker.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`EqualDivisionTimeTokenMaker.__repr__()`
Inherited from `timetokentools.TimeTokenMaker`

timetokentools.NoteFilledTimeTokenMaker

class `timetokentools.NoteFilledTimeTokenMaker`

New in version 2.8. Note-filled time-token maker:

```

>>> from abjad.tools import sequencetools
>>> from abjad.tools import timetokentools

>>> maker = timetokentools.NoteFilledTimeTokenMaker()

>>> duration_tokens = [(5, 16), (3, 8)]
>>> leaf_lists = maker(duration_tokens)
>>> leaves = sequencetools.flatten_sequence(leaf_lists)

>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens)
>>> staff = Staff(measures)
>>> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

>>> f(staff)
\new Staff {
  {

```

```

        \time 5/16
        c'4
        c'16
    }
    {
        \time 3/8
        c'4.
    }
}

```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`NoteFilledTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NoteFilledTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```

self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions

```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`NoteFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`NoteFilledTimeTokenMaker.__eq__(other)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`NoteFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NoteFilledTimeTokenMaker.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

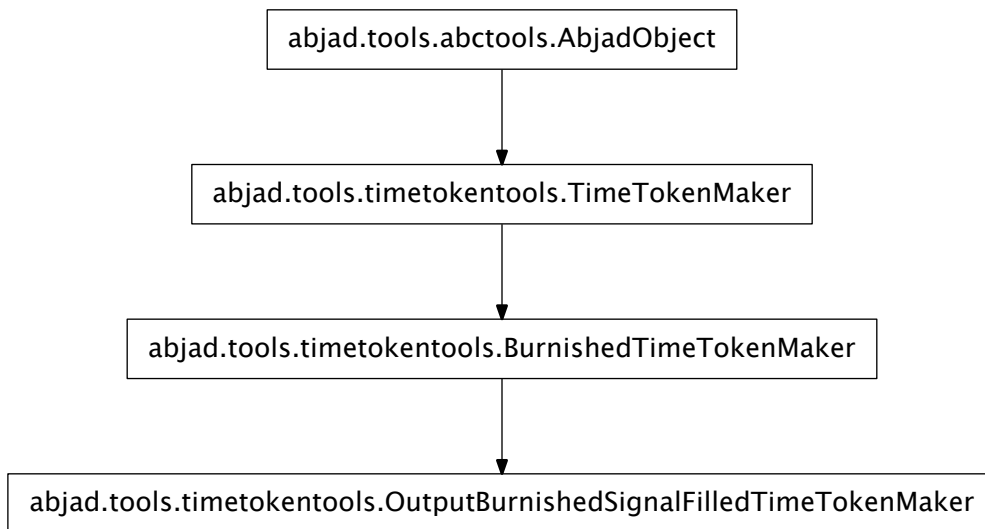
`NoteFilledTimeTokenMaker.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`NoteFilledTimeTokenMaker.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`NoteFilledTimeTokenMaker.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`NoteFilledTimeTokenMaker.__repr__()`

timetokentools.OutputBurnishedSignalFilledTimeTokenMaker



```
class timetokentools.OutputBurnishedSignalFilledTimeTokenMaker (pattern,      denom-
                                                                inator,      prola-
                                                                tion_addenda=None,
                                                                lefts=None,
                                                                middles=None,
                                                                rights=None,
                                                                left_lengths=None,
                                                                right_lengths=None,
                                                                sec-
                                                                ondary_divisions=None,
                                                                pat-
                                                                tern_helper=None,
                                                                prola-
                                                                tion_addenda_helper=None,
                                                                lefts_helper=None,
                                                                mid-
                                                                dles_helper=None,
                                                                rights_helper=None,
                                                                left_lengths_helper=None,
                                                                right_lengths_helper=None,
                                                                sec-
                                                                ondary_divisions_helper=None)
```

New in version 2.8. Output-burnished signal-filled time-token maker.

Configure the time-token maker at initialization:

```
>>> from abjad.tools import sequencetools
>>> from abjad.tools import timetokentools

>>> pattern, denominator, prolation_addenda = [1, 2, 3], 16, [0, 2]
>>> lefts, middles, rights = [-1], [0], [-1]
>>> left_lengths, right_lengths = [1], [1]
>>> secondary_divisions = [9]
>>> maker = timetokentools.OutputBurnishedSignalFilledTimeTokenMaker(
... pattern, denominator, prolation_addenda, lefts, middles, rights,
... left_lengths, right_lengths, secondary_divisions)
```

Then call the time-token maker on arbitrary duration tokens:

```
>>> duration_tokens = [(3, 8), (4, 8)]
>>> music = maker(duration_tokens)
```

The resulting Abjad objects can be included in any score:

```
>>> music = sequencetools.flatten_sequence(music)
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens)
>>> staff = Staff(measures)
>>> measures = measuretools.replace_contents_of_measures_in_expr(staff, music)

>>> f(staff)
\new Staff {
  {
    \time 3/8
    {
      r16
      c'8
      c'8.
    }
  }
}
```

```

    }
    {
        \time 4/8
        \fraction \times 3/5 {
            c'16
            c'8
            c'8
        }
    }
    {
        c'16
        c'16
        c'8
        r16
    }
}

```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`OutputBurnishedSignalFilledTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`OutputBurnishedSignalFilledTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```

self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions

```

Operate in place and return none.

Inherited from `timetoken.tools.TimeTokenMaker`

Special Methods

`OutputBurnishedSignalFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetoken.tools.BurnishedTimeTokenMaker`

`OutputBurnishedSignalFilledTimeTokenMaker.__eq__(other)`

Inherited from `timetoken.tools.BurnishedTimeTokenMaker`

`OutputBurnishedSignalFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputBurnishedSignalFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OutputBurnishedSignalFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputBurnishedSignalFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputBurnishedSignalFilledTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

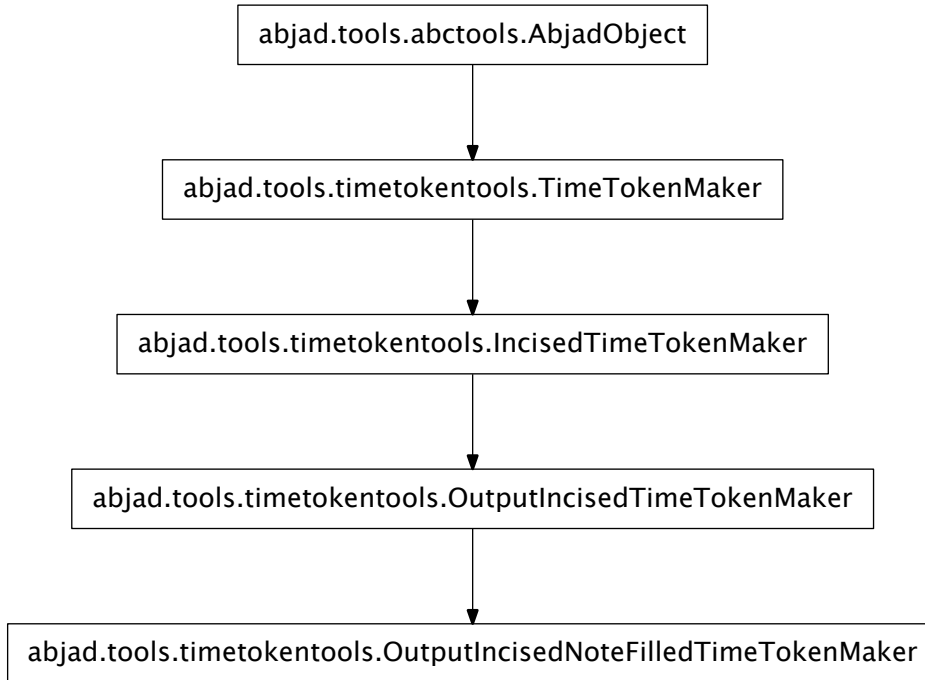
Return boolean.

Inherited from `abctools.AbjadObject`

`OutputBurnishedSignalFilledTimeTokenMaker.__repr__()`

Inherited from `timetokentools.TimeTokenMaker`

timetokentools.OutputIncisedNoteFilledTimeTokenMaker



```

class timetokentools.OutputIncisedNoteFilledTimeTokenMaker (prefix_signal,      pre-
                                                                fix_lengths, suffix_signal,
                                                                suffix_lengths, de-
                                                                nominator,      prola-
                                                                tion_addenda=None, sec-
                                                                ondary_divisions=None,
                                                                pre-
                                                                fix_signal_helper=None,
                                                                pre-
                                                                fix_lengths_helper=None,
                                                                suf-
                                                                fix_signal_helper=None,
                                                                suf-
                                                                fix_lengths_helper=None,
                                                                prola-
                                                                tion_addenda_helper=None,
                                                                sec-
                                                                ondary_divisions_helper=None)
  
```

New in version 2.8. Output-incised note-filled time-token maker.

Configure the time-token maker on initialization:

```

>>> from abjad.tools import sequencetools
>>> from abjad.tools import timetokentools
  
```

```
>>> prefix_signal, prefix_lengths = [-8], [2]
>>> suffix_signal, suffix_lengths = [-3], [4]
>>> denominator = 32
>>> maker = timetokentools.OutputIncisedNoteFilledTimeTokenMaker(
... prefix_signal, prefix_lengths, suffix_signal, suffix_lengths, denominator)
```

Then call the time-token maker on arbitrary duration tokens:

```
>>> duration_tokens = [(5, 8), (5, 8), (5, 8)]
>>> leaf_lists = maker(duration_tokens)
>>> leaves = sequencetools.flatten_sequence(leaf_lists)
```

The resulting Abjad objects can be included in any score and the time-token maker can be reused:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens)
>>> staff = Staff(measures)
>>> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

>>> f(staff)
\new Staff {
  {
    \time 5/8
    r4
    r4
    c'8
  }
  {
    c'2
    c'8
  }
  {
    c'4
    r16.
    r16.
    r16.
    r16.
  }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`OutputIncisedNoteFilledTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`OutputIncisedNoteFilledTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`OutputIncisedNoteFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedNoteFilledTimeTokenMaker.__eq__(other)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedNoteFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__ne__(arg)`

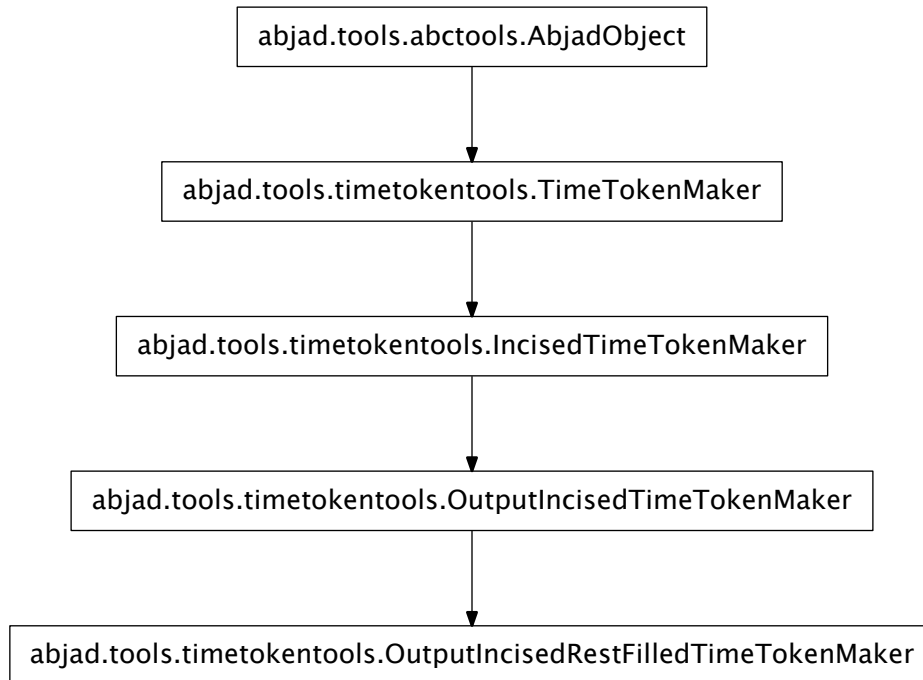
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__repr__()`

Inherited from `timetokentools.TimeTokenMaker`

timetokentools.OutputIncisedRestFilledTimeTokenMaker

```
class timetokentools.OutputIncisedRestFilledTimeTokenMaker (prefix_signal,      pre-  
                                                         fix_lengths, suffix_signal,  
                                                         suffix_lengths,      de-  
                                                         ominator,         prola-  
                                                         tion_addenda=None, sec-  
                                                         ondary_divisions=None,  
                                                         pre-  
                                                         fix_signal_helper=None,  
                                                         pre-  
                                                         fix_lengths_helper=None,  
                                                         suf-  
                                                         fix_signal_helper=None,  
                                                         suf-  
                                                         fix_lengths_helper=None,  
                                                         prola-  
                                                         tion_addenda_helper=None,  
                                                         sec-  
                                                         ondary_divisions_helper=None)
```

New in version 2.8. Output-incised rest-filled time-token maker.

Configure the time-token maker on initialization:

```
>>> from abjad.tools import sequencetools  
>>> from abjad.tools import timetokentools
```

```
>>> prefix_signal, prefix_lengths = [8], [2]
>>> suffix_signal, suffix_lengths = [3], [4]
>>> denominator = 32
>>> maker = timetokentools.OutputIncisedRestFilledTimeTokenMaker(
... prefix_signal, prefix_lengths, suffix_signal, suffix_lengths, denominator)
```

Then call the time-token maker on arbitrary duration tokens:

```
>>> duration_tokens = [(5, 8), (5, 8), (5, 8)]
>>> leaf_lists = maker(duration_tokens)
>>> leaves = sequencetools.flatten_sequence(leaf_lists)
```

The resulting Abjad objects can be included in any score and the time-token maker can be reused:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens)
>>> staff = Staff(measures)
>>> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

>>> f(staff)
\new Staff {
  {
    \time 5/8
    c'4
    c'4
    r8
  }
  {
    r2
    r8
  }
  {
    r4
    c'16.
    c'16.
    c'16.
    c'16.
  }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`OutputIncisedRestFilledTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`OutputIncisedRestFilledTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`OutputIncisedRestFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedRestFilledTimeTokenMaker.__eq__(other)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedRestFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

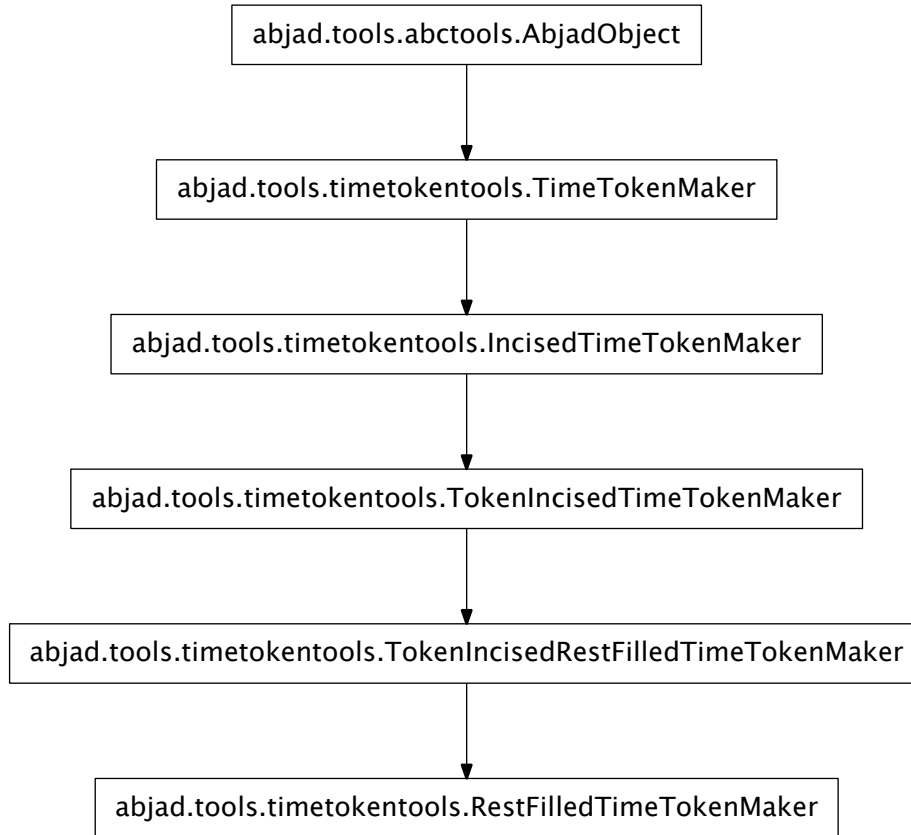
Return boolean.

Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__repr__()`

Inherited from `timetokentools.TimeTokenMaker`

timetokentools.RestFilledTimeTokenMaker



class timetokentools.**RestFilledTimeTokenMaker**

New in version 2.8. Rest-filled time-token maker:

```

>>> from abjad.tools import sequencetools
>>> from abjad.tools import timetokentools

>>> maker = timetokentools.RestFilledTimeTokenMaker()

>>> duration_tokens = [(5, 16), (3, 8)]
>>> leaf_lists = maker(duration_tokens)
>>> leaves = sequencetools.flatten_sequence(leaf_lists)

>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens)
>>> staff = Staff(measures)
>>> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

>>> f(staff)
\new Staff {
  {

```

```

        \time 5/16
        r4
        r16
    }
    {
        \time 3/8
        r4.
    }
}

```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`RestFilledTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`RestFilledTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```

self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions

```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`RestFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`RestFilledTimeTokenMaker.__eq__(other)`

Inherited from `timetokentools.IncisedTimeTokenMaker`

`RestFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RestFilledTimeTokenMaker.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

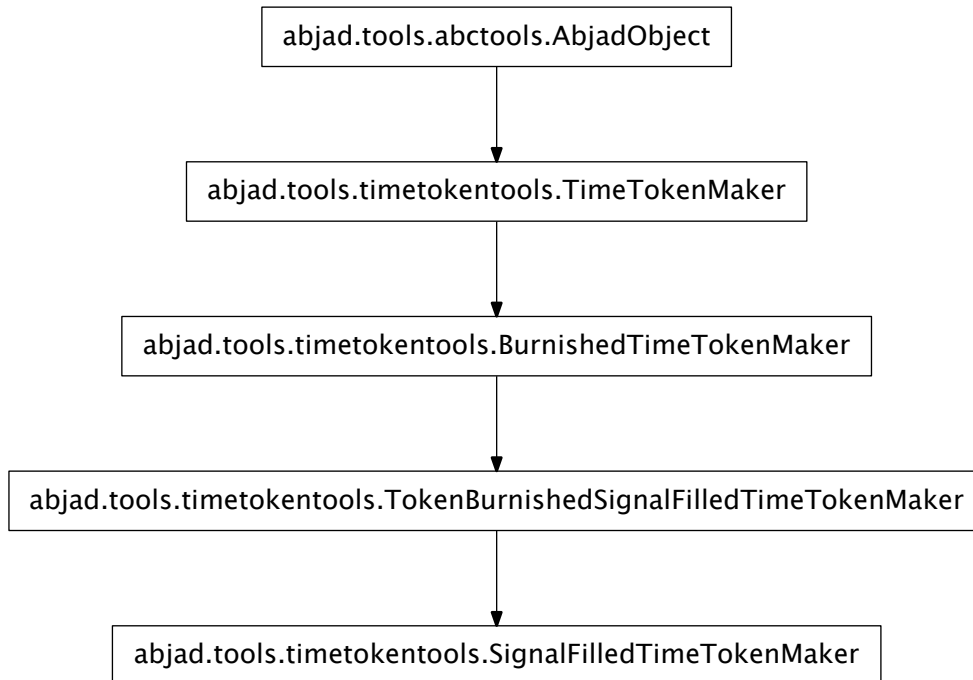
`RestFilledTimeTokenMaker.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`RestFilledTimeTokenMaker.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`RestFilledTimeTokenMaker.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`RestFilledTimeTokenMaker.__repr__()`
 Inherited from `timetokentools.TimeTokenMaker`

timetokentools.SignalFilledTimeTokenMaker



```

class timetokentools.SignalFilledTimeTokenMaker (pattern,          denominator,          pro-
                                                    lation_addenda=None,          sec-
                                                    ondary_divisions=None,          pat-
                                                    tern_helper=None,          prola-
                                                    tion_addenda_helper=None,          sec-
                                                    ondary_divisions_helper=None)

```

New in version 2.8. Signal-affixed time-token maker.

Configure the time-token maker at initialization:

```

>>> from abjad.tools import sequencetools
>>> from abjad.tools import timetokentools

>>> pattern, denominator, prolation_addenda = [-1, 4, -2, 3], 16, [3, 4]
>>> maker = timetokentools.SignalFilledTimeTokenMaker(
...     pattern, denominator, prolation_addenda)

```

Then call the time-token maker on arbitrary duration tokens:

```

>>> duration_tokens = [(2, 8), (5, 8)]
>>> music = maker(duration_tokens)

```

The resulting Abjad objects can be included in any score and the time-token make can be called indefinitely on other arbitrary sequences of duration tokens:

```
>>> music = sequencetools.flatten_sequence(music)
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens)
>>> staff = Staff(measures)
>>> measures = measuretools.replace_contents_of_measures_in_expr(staff, music)

>>> f(staff)
\new Staff {
  {
    \time 2/8
    \times 4/7 {
      r16
      c'4
      r8
    }
  }
  {
    \time 5/8
    \fraction \times 5/7 {
      c'8.
      r16
      c'4
      r8
      c'8.
      r16
    }
  }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`SignalFilledTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`SignalFilledTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`SignalFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetokentools.BurnishedTimeTokenMaker`

`SignalFilledTimeTokenMaker.__eq__(other)`

Inherited from `timetokentools.BurnishedTimeTokenMaker`

`SignalFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

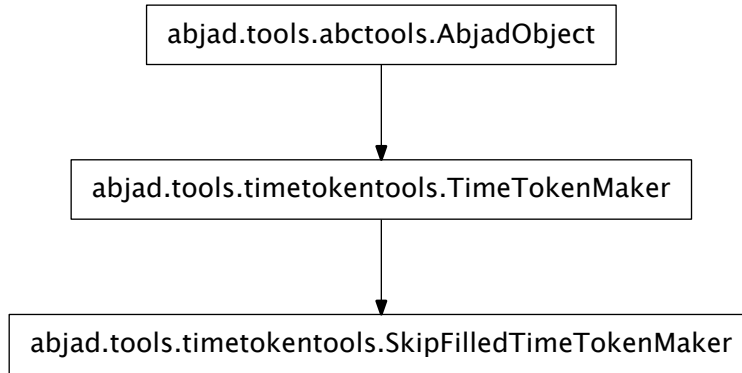
Return boolean.

Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__repr__()`

Inherited from `timetokentools.TimeTokenMaker`

timetokentools.SkipFilledTimeTokenMaker



class timetokentools.**SkipFilledTimeTokenMaker**

New in version 2.10. Skip-filled time-token maker:

```

>>> maker = timetokentools.SkipFilledTimeTokenMaker()

>>> duration_tokens = [(1, 5), (1, 4), (1, 6), (7, 9)]
>>> leaf_lists = maker(duration_tokens)
>>> leaves = sequencetools.flatten_sequence(leaf_lists)

>>> staff = Staff(leaves)

>>> f(staff)
\new Staff {
  s1 * 1/5
  s1 * 1/4
  s1 * 1/6
  s1 * 7/9
}
  
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`SkipFilledTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`SkipFilledTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetoken.tools.TimeTokenMaker`

Special Methods

`SkipFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

`SkipFilledTimeTokenMaker.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SkipFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SkipFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SkipFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SkipFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SkipFilledTimeTokenMaker.__ne__(arg)`

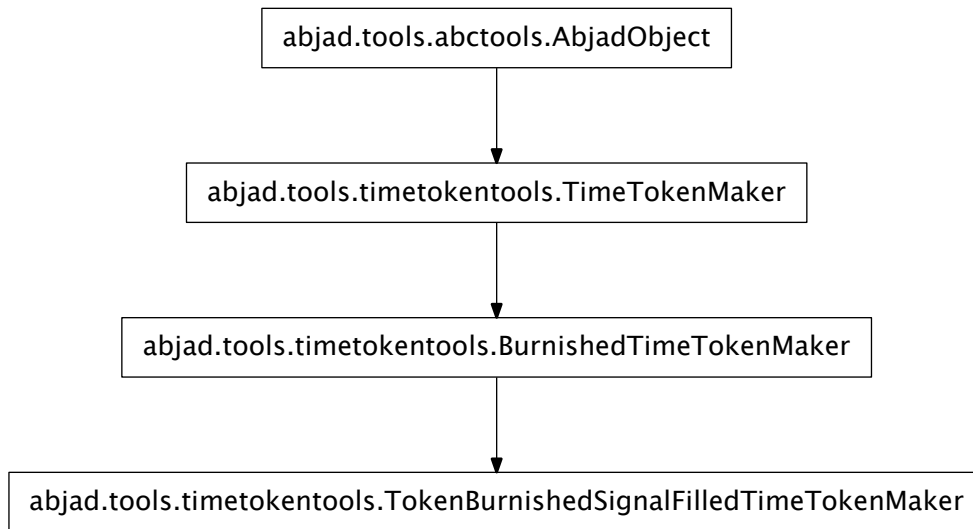
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SkipFilledTimeTokenMaker.__repr__()`
 Inherited from `timetokenmaker.TimeTokenMaker`

timetokenmaker.TokenBurnishedSignalFilledTimeTokenMaker



```

class timetokenmaker.TokenBurnishedSignalFilledTimeTokenMaker (pattern,      denom-
                                                                inator,      prola-
                                                                tion_addenda=None,
                                                                lefts=None,    mid-
                                                                dles=None,
                                                                rights=None,
                                                                left_lengths=None,
                                                                right_lengths=None,
                                                                sec-
                                                                ondary_divisions=None,
                                                                pat-
                                                                tern_helper=None,
                                                                prola-
                                                                tion_addenda_helper=None,
                                                                lefts_helper=None,
                                                                mid-
                                                                dles_helper=None,
                                                                rights_helper=None,
                                                                left_lengths_helper=None,
                                                                right_lengths_helper=None,
                                                                sec-
                                                                ondary_divisions_helper=None)
    
```

New in version 2.8. Token-burnished signal-filled time-token maker.
 Configure the time-token maker at instantiation:

```
>>> from abjad.tools import sequencetools
>>> from abjad.tools import timetokentools

>>> pattern, denominator, prolation_addenda = [1, 1, 2, 4], 32, [0, 3]
>>> lefts, middles, rights = [-1], [0], [-1]
>>> left_lengths, right_lengths = [1], [1]
>>> secondary_divisions = [14]
>>> maker = timetokentools.TokenBurnishedSignalFilledTimeTokenMaker(
... pattern, denominator, prolation_addenda, lefts, middles, rights,
... left_lengths, right_lengths, secondary_divisions)
```

Then call the time-token maker on any sequence of duration tokens:

```
>>> duration_tokens = [(5, 16), (6, 16)]
>>> music = maker(duration_tokens)
```

The resulting Abjad objects can then be included in any score and the time-token maker can be called again and again on different duration tokens:

```
>>> music = sequencetools.flatten_sequence(music)
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens)
>>> staff = Staff(measures)
>>> measures = measuretools.replace_contents_of_measures_in_expr(staff, music)

>>> f(staff)
\new Staff {
  {
    \time 5/16
    {
      r32
      c'32
      c'16
      c'8
      c'32
      r32
    }
  }
  {
    \time 6/16
    \times 4/7 {
      r16
      c'8
      r32
    }
    {
      r32
      c'16
      c'8
      r32
    }
  }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`TokenBurnishedSignalFilledTimeTokenMaker.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TokenBurnishedSignalFilledTimeTokenMaker.reverse()`
New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetoken tools.TimeTokenMaker`

Special Methods

`TokenBurnishedSignalFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetoken tools.BurnishedTimeTokenMaker`

`TokenBurnishedSignalFilledTimeTokenMaker.__eq__(other)`

Inherited from `timetoken tools.BurnishedTimeTokenMaker`

`TokenBurnishedSignalFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

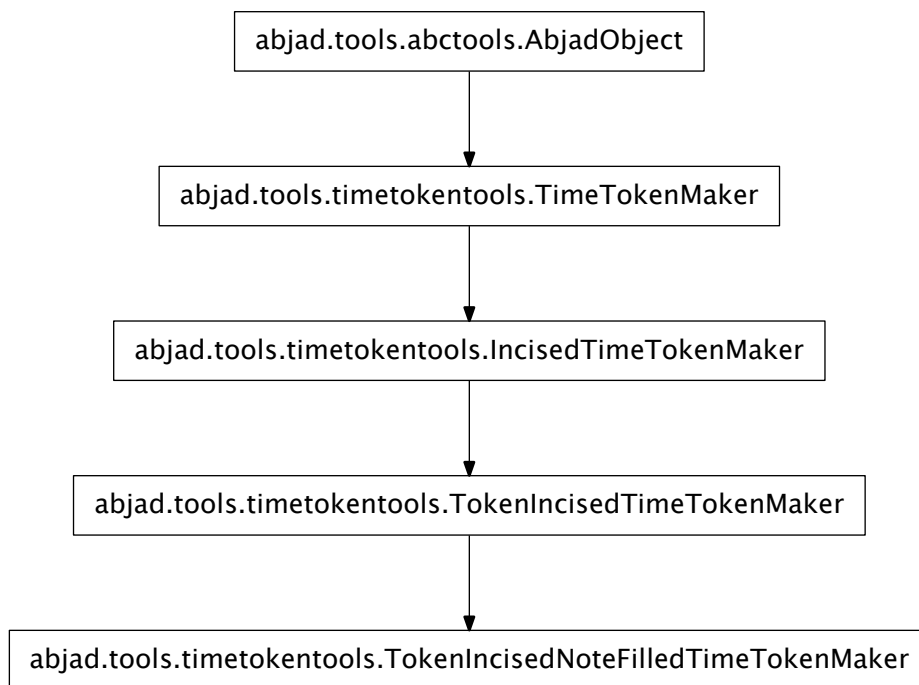
Return boolean.

Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__repr__()`

Inherited from `timetokentools.TimeTokenMaker`

timetokentools.TokenIncisedNoteFilledTimeTokenMaker



```
class timetokentools.TokenIncisedNoteFilledTimeTokenMaker (prefix_signal,      pre-
                                                             fix_lengths,  suffix_signal,
                                                             suffix_lengths,  de-
                                                             nominator,      prola-
                                                             tion_addenda=None,  sec-
                                                             ondary_divisions=None,
                                                             pre-
                                                             fix_signal_helper=None,
                                                             pre-
                                                             fix_lengths_helper=None,
                                                             suf-
                                                             fix_signal_helper=None,
                                                             suf-
                                                             fix_lengths_helper=None,
                                                             prola-
                                                             tion_addenda_helper=None,
                                                             sec-
                                                             ondary_divisions_helper=None)
```

New in version 2.8. Token-incised note-filled time-token maker:

```
>>> from abjad.tools import sequencetools
>>> from abjad.tools import timetokentools

>>> prefix_signal, prefix_lengths = [-8], [0, 1]
>>> suffix_signal, suffix_lengths = [-1], [1]
>>> denominator = 32
>>> maker = timetokentools.TokenIncisedNoteFilledTimeTokenMaker(
... prefix_signal, prefix_lengths, suffix_signal, suffix_lengths, denominator)

>>> duration_tokens = [(5, 8), (5, 8), (5, 8), (5, 8)]
>>> leaf_lists = maker(duration_tokens)
>>> leaves = sequencetools.flatten_sequence(leaf_lists)

>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens)
>>> staff = Staff(measures)
>>> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

>>> f(staff)
\new Staff {
  {
    \time 5/8
    c'2
    c'16.
    r32
  }
  {
    r4
    c'4
    c'16.
    r32
  }
  {
    c'2
    c'16.
    r32
  }
}
```

```

        r4
        c'4
        c'16.
        r32
    }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`TokenIncisedNoteFilledTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TokenIncisedNoteFilledTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```

self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetoken.tools.TimeTokenMaker`

Special Methods

`TokenIncisedNoteFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetoken.tools.IncisedTimeTokenMaker`

`TokenIncisedNoteFilledTimeTokenMaker.__eq__(other)`

Inherited from `timetoken.tools.IncisedTimeTokenMaker`

`TokenIncisedNoteFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

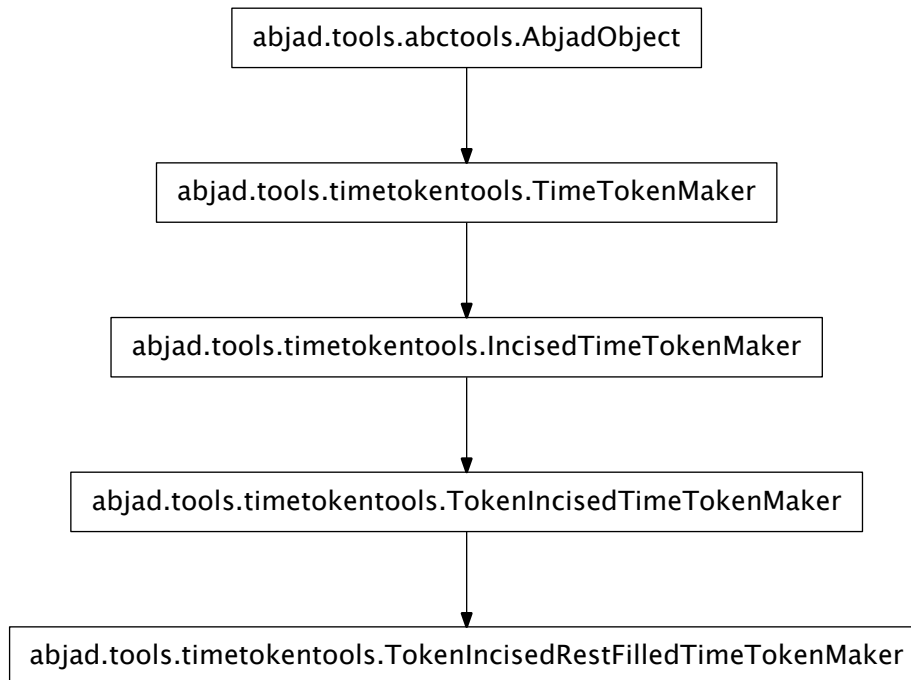
`TokenIncisedNoteFilledTimeTokenMaker.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__repr__()`
 Inherited from `timetoken.tools.TimeTokenMaker`

timetoken.tools.TokenIncisedRestFilledTimeTokenMaker



```
class timetokentools.TokenIncisedRestFilledTimeTokenMaker (prefix_signal,      pre-
                                                             fix_lengths,  suffix_signal,
                                                             suffix_lengths,  de-
                                                             nominator,      prola-
                                                             tion_addenda=None,  sec-
                                                             ondary_divisions=None,
                                                             pre-
                                                             fix_signal_helper=None,
                                                             pre-
                                                             fix_lengths_helper=None,
                                                             suf-
                                                             fix_signal_helper=None,
                                                             suf-
                                                             fix_lengths_helper=None,
                                                             prola-
                                                             tion_addenda_helper=None,
                                                             sec-
                                                             ondary_divisions_helper=None)
```

New in version 2.8. Token-incised rest-filled time-token maker:

```
>>> from abjad.tools import sequencetools
>>> from abjad.tools import timetokentools

>>> prefix_signal, prefix_lengths = [8], [1, 2, 3, 4]
>>> suffix_signal, suffix_lengths = [1], [1]
>>> denominator = 32
>>> maker = timetokentools.TokenIncisedRestFilledTimeTokenMaker(
... prefix_signal, prefix_lengths, suffix_signal, suffix_lengths, denominator)

>>> duration_tokens = [(5, 8), (5, 8), (5, 8), (5, 8)]
>>> leaf_lists = maker(duration_tokens)
>>> leaves = sequencetools.flatten_sequence(leaf_lists)

>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens)
>>> staff = Staff(measures)
>>> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

>>> f(staff)
\new Staff {
  {
    \time 5/8
    c'4
    r4
    r16.
    c'32
  }
  {
    c'4
    c'4
    r16.
    c'32
  }
  {
    c'4
    c'4
    c'8
  }
}
```

```
{
    c'4
    c'4
    c'8
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`TokenIncisedRestFilledTimeTokenMaker.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TokenIncisedRestFilledTimeTokenMaker.reverse()`

New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetoken.tools.TimeTokenMaker`

Special Methods

`TokenIncisedRestFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Inherited from `timetoken.tools.IncisedTimeTokenMaker`

`TokenIncisedRestFilledTimeTokenMaker.__eq__(other)`

Inherited from `timetoken.tools.IncisedTimeTokenMaker`

`TokenIncisedRestFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

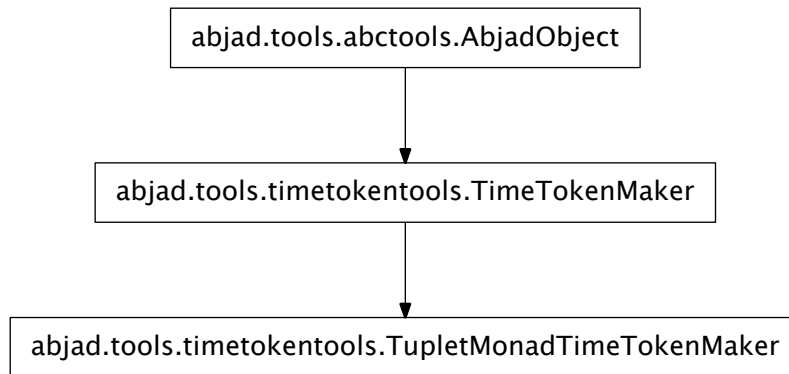
`TokenIncisedRestFilledTimeTokenMaker.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__repr__()`
 Inherited from `timetoken.tools.TimeTokenMaker`

`timetoken.tools.TupletMonadTimeTokenMaker`



class `timetoken.tools.TupletMonadTimeTokenMaker`

New in version 2.10. Tuplet monad time-token maker:

```

>>> maker = timetoken.tools.TupletMonadTimeTokenMaker()

>>> duration_tokens = [(1, 5), (1, 4), (1, 6), (7, 9)]
>>> tuplet_lists = maker(duration_tokens)
>>> tuplets = sequencetools.flatten_sequence(tuplet_lists)

>>> staff = Staff(tuplets)
  
```



```
>>> f(staff)
\new Staff {
  \times 4/5 {
    c'4
  }
  {
    c'4
  }
  \times 2/3 {
    c'4
  }
  \times 8/9 {
    c'2..
  }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Read-only Properties

`TupletMonadTimeTokenMaker.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`TupletMonadTimeTokenMaker.reverse()`
New in version 2.10. Reverse time token maker.

Note: method is provisional.

Defined equal to the in-place reversal of the following:

```
self.pattern
self.prolation_addenda
self.lefts
self.middles
self.rights
self.left_lengths
self.right_lengths
self.secondary_divisions
```

Operate in place and return none.

Inherited from `timetokentools.TimeTokenMaker`

Special Methods

`TupletMonadTimeTokenMaker.__call__(duration_tokens, seeds=None)`

`TupletMonadTimeTokenMaker.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`TupletMonadTimeTokenMaker.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`TupletMonadTimeTokenMaker.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`TupletMonadTimeTokenMaker.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`TupletMonadTimeTokenMaker.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

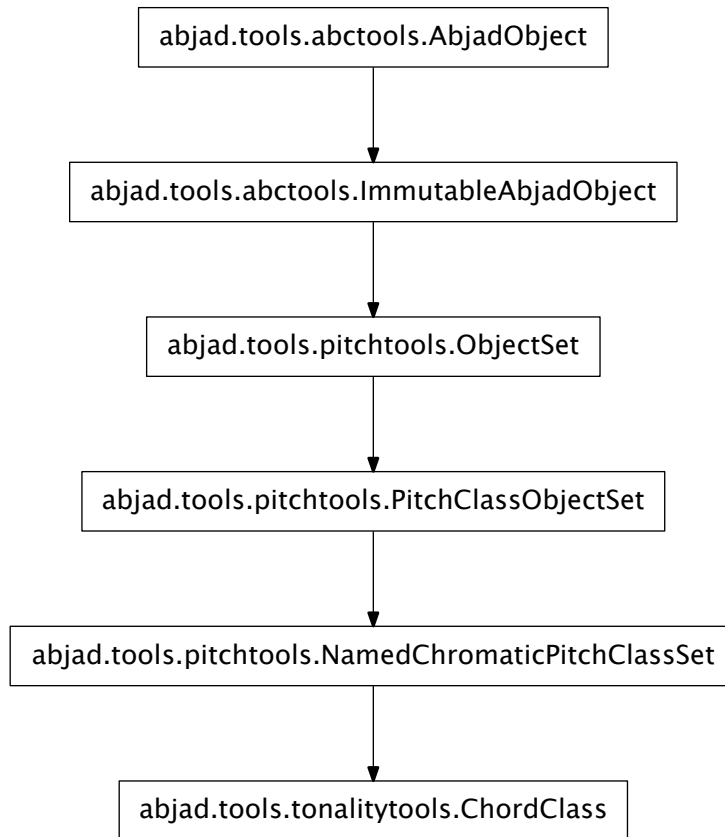
`TupletMonadTimeTokenMaker.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`TupletMonadTimeTokenMaker.__repr__()`
Inherited from `timetokentools.TimeTokenMaker`

tonalitytools

concrete classes

tonalitytools.ChordClass



class tonalitytools.**ChordClass** (*args, **kwargs)

New in version 2.0. Abjad model of tonal chords like G 7, G 6/5, G half-diminished 6/5, etc.

Note that notions like G 7 represent an entire *class of* chords because there are many different spacings and registrations of a G 7 chord.

Read-only Properties

ChordClass.**bass**

ChordClass.**cardinality**

ChordClass.**extent**

ChordClass.**figured_bass**

ChordClass.**inversion**

ChordClass.inversion_equivalent_diatonic_interval_class_vector

Inherited from `pitchtools.NamedChromaticPitchClassSet`

ChordClass.markup

ChordClass.named_chromatic_pitch_classes

Read-only named chromatic pitch-classes:

```
>>> named_chromatic_pitch_class_set = pitchtools.NamedChromaticPitchClassSet(
...     ['gs', 'g', 'as', 'c', 'cs'])
>>> for x in named_chromatic_pitch_class_set.named_chromatic_pitch_classes: x
...
NamedChromaticPitchClass('as')
NamedChromaticPitchClass('c')
NamedChromaticPitchClass('cs')
NamedChromaticPitchClass('g')
NamedChromaticPitchClass('gs')
```

Return tuple.

Inherited from `pitchtools.NamedChromaticPitchClassSet`

ChordClass.numbered_chromatic_pitch_class_set

Inherited from `pitchtools.NamedChromaticPitchClassSet`

ChordClass.quality_indicator

ChordClass.quality_pair

ChordClass.root

ChordClass.root_string

ChordClass.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

ChordClass.copy()

Return a shallow copy of a set.

Inherited from `__builtin__.frozenset`

ChordClass.difference()

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Inherited from `__builtin__.frozenset`

ChordClass.intersection()

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Inherited from `__builtin__.frozenset`

ChordClass.isdisjoint()

Return True if two sets have a null intersection.

Inherited from `__builtin__.frozenset`

`ChordClass.issubset()`
 Report whether another set contains this set.
 Inherited from `__builtin__.frozenset`

`ChordClass.issuperset()`
 Report whether this set contains another set.
 Inherited from `__builtin__.frozenset`

`ChordClass.order_by(npc_seg)`
 Inherited from `pitchtools.NamedChromaticPitchClassSet`

`ChordClass.symmetric_difference()`
 Return the symmetric difference of two sets as a new set.
 (i.e. all elements that are in exactly one of the sets.)
 Inherited from `__builtin__.frozenset`

`ChordClass.transpose()`

`ChordClass.union()`
 Return the union of sets as a new set.
 (i.e. all elements that are in either set.)
 Inherited from `__builtin__.frozenset`

Special Methods

`ChordClass.__and__()`
 $x._and_(y) \iff x \& y$
 Inherited from `__builtin__.frozenset`

`ChordClass.__cmp__(y) $\iff cmp(x, y)$`
 Inherited from `__builtin__.frozenset`

`ChordClass.__contains__()`
 $x._contains_(y) \iff y \text{ in } x$.
 Inherited from `__builtin__.frozenset`

`ChordClass.__eq__(arg)`

`ChordClass.__ge__()`
 $x._ge_(y) \iff x \geq y$
 Inherited from `__builtin__.frozenset`

`ChordClass.__gt__()`
 $x._gt_(y) \iff x > y$
 Inherited from `__builtin__.frozenset`

`ChordClass.__hash__()`
 Inherited from `pitchtools.NamedChromaticPitchClassSet`

`ChordClass.__iter__()` $\iff iter(x)$
 Inherited from `__builtin__.frozenset`

`ChordClass.__le__()`
 $x._le_(y) \iff x \leq y$
 Inherited from `__builtin__.frozenset`

```

ChordClass.__len__() <==> len(x)
    Inherited from __builtin__.frozenset

ChordClass.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.frozenset

ChordClass.__ne__(arg)

ChordClass.__or__()
    x.__or__(y) <==> x|y
    Inherited from __builtin__.frozenset

ChordClass.__rand__()
    x.__rand__(y) <==> y&x
    Inherited from __builtin__.frozenset

ChordClass.__repr__()

ChordClass.__ror__()
    x.__ror__(y) <==> y|x
    Inherited from __builtin__.frozenset

ChordClass.__rsub__()
    x.__rsub__(y) <==> y-x
    Inherited from __builtin__.frozenset

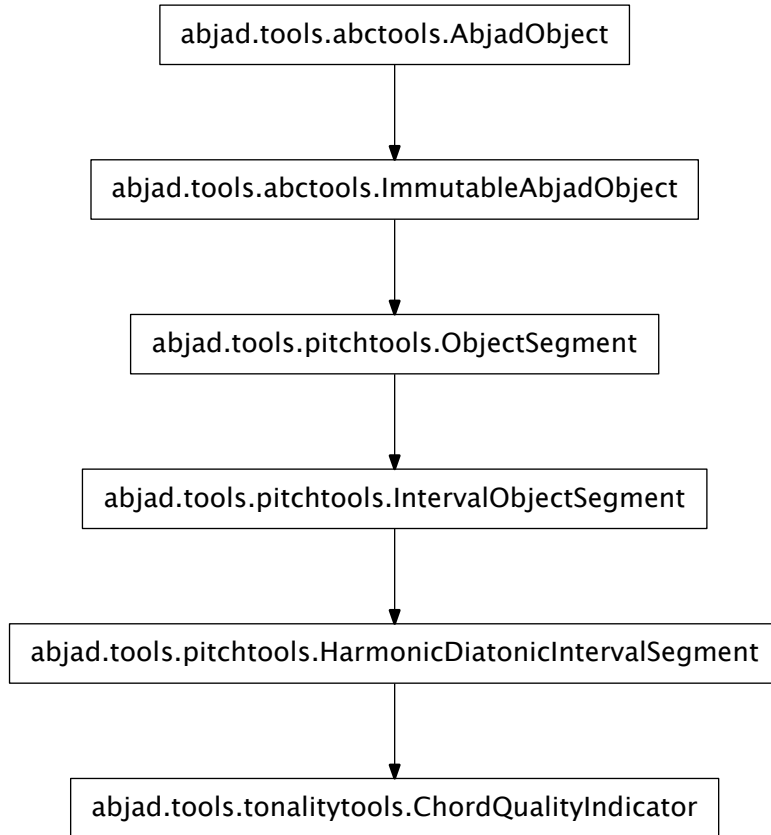
ChordClass.__rxor__()
    x.__rxor__(y) <==> y^x
    Inherited from __builtin__.frozenset

ChordClass.__str__()
    Inherited from pitchtools.NamedChromaticPitchClassSet

ChordClass.__sub__()
    x.__sub__(y) <==> x-y
    Inherited from __builtin__.frozenset

ChordClass.__xor__()
    x.__xor__(y) <==> x^y
    Inherited from __builtin__.frozenset

```

tonalitytools.ChordQualityIndicator

class `tonalitytools.ChordQualityIndicator` (*args, **kwargs)
 New in version 2.0. Chord quality indicator.

Read-only Properties

`ChordQualityIndicator.cardinality`

`ChordQualityIndicator.extent`

`ChordQualityIndicator.extent_name`

`ChordQualityIndicator.harmonic_chromatic_interval_segment`

Inherited from `pitchtools.HarmonicDiatonicIntervalSegment`

`ChordQualityIndicator.interval_classes`

Inherited from `pitchtools.IntervalObjectSegment`

`ChordQualityIndicator.intervals`

Inherited from `pitchtools.IntervalObjectSegment`

`ChordQualityIndicator.inversion`

`ChordQualityIndicator.melodic_chromatic_interval_segment`

Inherited from `pitchtools.HarmonicDiatonicIntervalSegment`

`ChordQualityIndicator.melodic_diatonic_interval_segment`

Inherited from `pitchtools.HarmonicDiatonicIntervalSegment`

`ChordQualityIndicator.position`

`ChordQualityIndicator.quality_string`

`ChordQualityIndicator.rotation`

`ChordQualityIndicator.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`ChordQualityIndicator.count(value) → integer` – return number of occurrences of value

Inherited from `__builtin__.tuple`

`ChordQualityIndicator.index(value[, start[, stop]]) → integer` – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`ChordQualityIndicator.rotate(n)`

Inherited from `pitchtools.IntervalObjectSegment`

Special Methods

`ChordQualityIndicator.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`ChordQualityIndicator.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__getslice__(start, stop)`

Inherited from `pitchtools.ObjectSegment`

`ChordQualityIndicator.__gt__()`

`x.__gt__(y) <==> x>y`

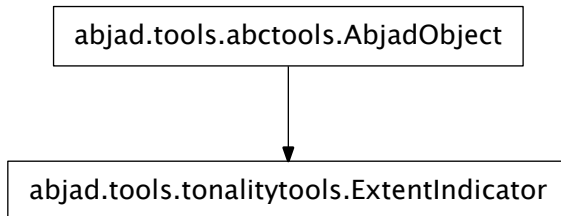
Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__hash__() <==> hash(x)`

Inherited from `__builtin__.tuple`


```
ChordQualityIndicator.__iter__() <==> iter(x)
    Inherited from __builtin__.tuple
ChordQualityIndicator.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.tuple
ChordQualityIndicator.__len__() <==> len(x)
    Inherited from __builtin__.tuple
ChordQualityIndicator.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple
ChordQualityIndicator.__mul__(n)
    Inherited from pitchtools.ObjectSegment
ChordQualityIndicator.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple
ChordQualityIndicator.__repr__()
ChordQualityIndicator.__rmul__(n)
    Inherited from pitchtools.ObjectSegment
ChordQualityIndicator.__str__()
    Inherited from pitchtools.IntervalObjectSegment
```

tonalitytools.ExtentIndicator



class `tonalitytools.ExtentIndicator` (*arg*)
 New in version 2.0. Indicator of chord extent, such as triad, seventh chord, ninth chord, etc.
 Value object that can not be changed after instantiation.

Read-only Properties

```
ExtentIndicator.name
ExtentIndicator.number
```

`ExtentIndicator.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ExtentIndicator.__eq__(arg)`

`ExtentIndicator.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ExtentIndicator.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ExtentIndicator.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ExtentIndicator.__lt__(arg)`

Abjad objects by default do not implement this method.

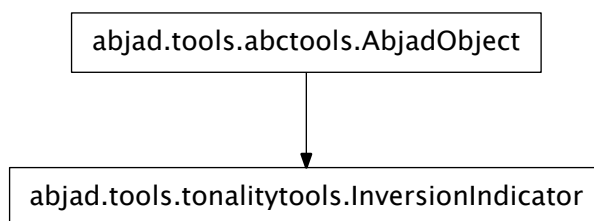
Raise exception.

Inherited from `abctools.AbjadObject`

`ExtentIndicator.__ne__(arg)`

`ExtentIndicator.__repr__()`

`tonalitytools.InversionIndicator`



class `tonalitytools.InversionIndicator` (*arg=0*)

New in version 2.0. Indicator of the inversion of tertian chords: 5, 63, 64 and also 7, 65, 43, 42, etc. Also root position, first, second, third inversions, etc.

Value object that can not be changed once initialized.

Read-only Properties

`InversionIndicator.name`

`InversionIndicator.number`

`InversionIndicator.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`InversionIndicator.title`

Methods

`InversionIndicator.extent_to_figured_bass_string` (*extent*)

Special Methods

`InversionIndicator.__eq__` (*arg*)

`InversionIndicator.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InversionIndicator.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`InversionIndicator.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`InversionIndicator.__lt__` (*arg*)

Abjad objects by default do not implement this method.

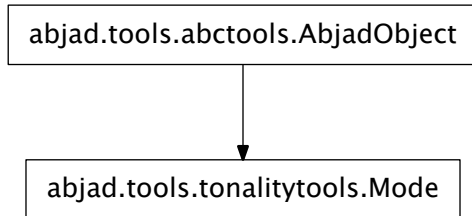
Raise exception.

Inherited from `abctools.AbjadObject`

`InversionIndicator.__ne__` (*arg*)

`InversionIndicator.__repr__` ()

tonalitytools.Mode



class `tonalitytools.Mode` (*arg*)

New in version 2.0. Diatonic mode. Can be extended for nondiatonic mode.

Modes with different ascending and descending forms not yet implemented.

Read-only Properties

`Mode.melodic_diatonic_interval_segment`

`Mode.mode_name`

`Mode.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`Mode.__eq__` (*arg*)

`Mode.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Mode.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Mode.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Mode.__len__` ()

`Mode.__lt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

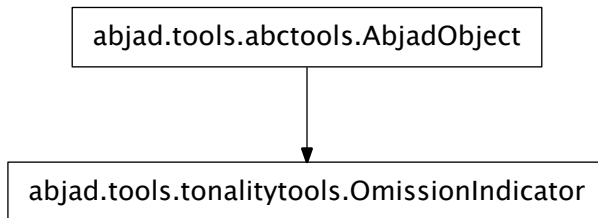
Inherited from `abctools.AbjadObject`

`Mode.__ne__(arg)`

`Mode.__repr__()`

`Mode.__str__()`

`tonalitytools.OmissionIndicator`



class `tonalitytools.OmissionIndicator`

New in version 2.0. Indicator of missing chord tones.

Value object that can not be chnaged after instantiation.

Read-only Properties

`OmissionIndicator.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`OmissionIndicator.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`OmissionIndicator.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OmissionIndicator.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

OmissionIndicator.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

OmissionIndicator.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

OmissionIndicator.**__ne__**(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

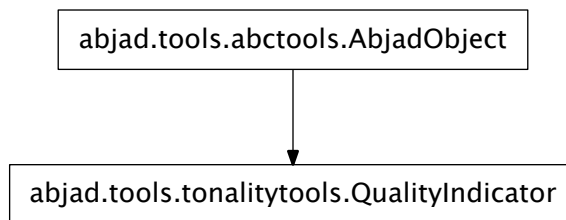
OmissionIndicator.**__repr__**()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

tonalitytools.QualityIndicator



class `tonalitytools.QualityIndicator`(quality_string)

New in version 2.0. Indicator of chord quality, such as major, minor, dominant, diminished, etc.

Value object that can not be changed after instantiation.

Read-only Properties

`QualityIndicator.is_uppercase`

`QualityIndicator.quality_string`

`QualityIndicator.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`QualityIndicator.__eq__(arg)`

`QualityIndicator.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`QualityIndicator.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`QualityIndicator.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`QualityIndicator.__lt__(arg)`

Abjad objects by default do not implement this method.

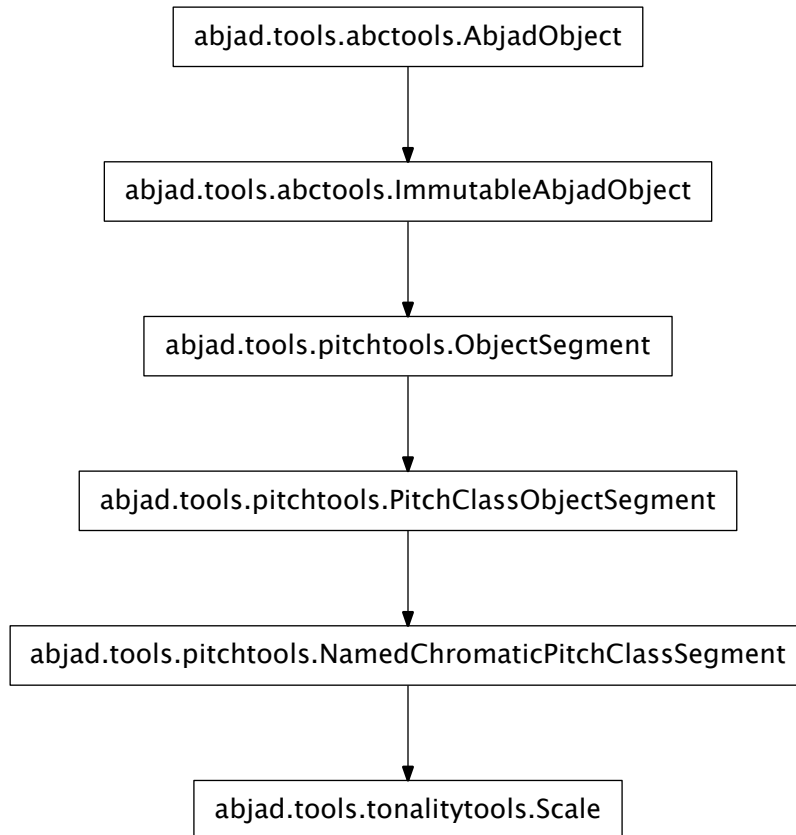
Raise exception.

Inherited from `abctools.AbjadObject`

`QualityIndicator.__ne__(arg)`

`QualityIndicator.__repr__()`

tonalitytools.Scale



```
class tonalitytools.Scale (*args, **kwargs)
    New in version 2.0. Abjad model of diatonic scale.
```

Read-only Properties

Scale.**diatonic_interval_class_segment**

Scale.**dominant**

Scale.**inversion_equivalent_diatonic_interval_class_segment**

Inherited from pitchtools.NamedChromaticPitchClassSegment

Scale.**key_signature**

Scale.**leading_tone**

Scale.**mediant**

Scale.**named_chromatic_pitch_class_set**

Inherited from pitchtools.NamedChromaticPitchClassSegment

`Scale.named_chromatic_pitch_classes`

Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.numbered_chromatic_pitch_class_segment`

Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.numbered_chromatic_pitch_class_set`

Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.numbered_chromatic_pitch_classes`

Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`Scale.subdominant`

`Scale.submediant`

`Scale.superdominant`

`Scale.tonic`

Methods

`Scale.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.tuple`

`Scale.create_named_chromatic_pitch_set_in_pitch_range(pitch_range)`

`Scale.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.tuple`

`Scale.is_equivalent_under_transposition(arg)`

Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.named_chromatic_pitch_class_to_scale_degree(*args)`

`Scale.retrograde()`

Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.rotate(n)`

Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.scale_degree_to_named_chromatic_pitch_class(*args)`

`Scale.transpose(melodic_diatonic_interval)`

Inherited from `pitchtools.NamedChromaticPitchClassSegment`

Special Methods

`Scale.__add__(arg)`

Inherited from `pitchtools.ObjectSegment`

`Scale.__contains__()`

`x.__contains__(y) <==> y in x`

Inherited from `__builtin__.tuple`

```

Scale.__eq__()
    x.__eq__(y) <==> x==y
    Inherited from __builtin__.tuple

Scale.__ge__()
    x.__ge__(y) <==> x>=y
    Inherited from __builtin__.tuple

Scale.__getitem__()
    x.__getitem__(y) <==> x[y]
    Inherited from __builtin__.tuple

Scale.__getslice__(start, stop)
    Inherited from pitchtools.ObjectSegment

Scale.__gt__()
    x.__gt__(y) <==> x>y
    Inherited from __builtin__.tuple

Scale.__hash__() <==> hash(x)
    Inherited from __builtin__.tuple

Scale.__iter__() <==> iter(x)
    Inherited from __builtin__.tuple

Scale.__le__()
    x.__le__(y) <==> x<=y
    Inherited from __builtin__.tuple

Scale.__len__() <==> len(x)
    Inherited from __builtin__.tuple

Scale.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.tuple

Scale.__mul__(n)
    Inherited from pitchtools.ObjectSegment

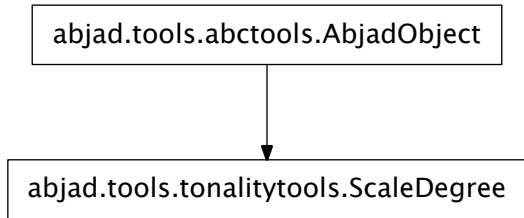
Scale.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.tuple

Scale.__repr__()

Scale.__rmul__(n)
    Inherited from pitchtools.ObjectSegment

Scale.__str__()
    Inherited from pitchtools.NamedChromaticPitchClassSegment

```

tonalitytools.ScaleDegree

class `tonalitytools.ScaleDegree` (*args)

New in version 2.0. Abjad model of diatonic scale degrees 1, 2, 3, 4, 5, 6, 7 and also chromatic alterations including flat-2, flat-3, flat-6, etc.

Read-only Properties

`ScaleDegree.accidental`

Read-only accidental applied to scale degree.

`ScaleDegree.name`

Read-only name of scale degree.

`ScaleDegree.number`

Read-only number of diatonic scale degree from 1 to 7, inclusive.

`ScaleDegree.roman_numeral_string`

`ScaleDegree.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`ScaleDegree.symbolic_string`

`ScaleDegree.title_string`

Methods

`ScaleDegree.apply_accidental` (*accidental*)

Apply accidental to self and emit new instance.

Special Methods

`ScaleDegree.__eq__` (*arg*)

`ScaleDegree.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ScaleDegree.__gt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ScaleDegree.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ScaleDegree.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

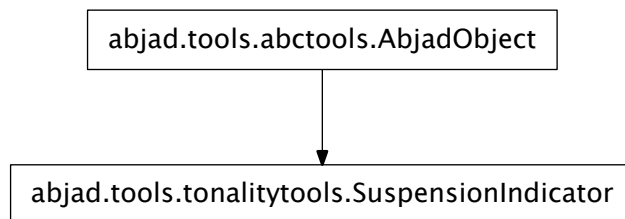
Inherited from `abctools.AbjadObject`

`ScaleDegree.__ne__(arg)`

`ScaleDegree.__repr__()`

`ScaleDegree.__str__()`

tonalitytools.SuspensionIndicator



class `tonalitytools.SuspensionIndicator(*args)`

New in version 2.0. Indicator of 9-8, 7-6, 4-3, 2-1 and other types of suspension typical of, for example, the Bach chorales.

Value object that can not be changed after instantiation.

Read-only Properties

`SuspensionIndicator.chord_name`

`SuspensionIndicator.figured_bass_pair`

`SuspensionIndicator.figured_bass_string`

`SuspensionIndicator.is_empty`

`SuspensionIndicator.start`

`SuspensionIndicator.stop`

`SuspensionIndicator.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`SuspensionIndicator.title_string`

Special Methods

`SuspensionIndicator.__eq__(arg)`

`SuspensionIndicator.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SuspensionIndicator.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SuspensionIndicator.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SuspensionIndicator.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

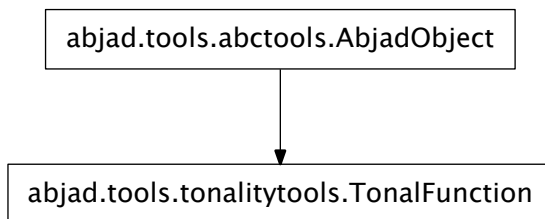
Inherited from `abctools.AbjadObject`

`SuspensionIndicator.__ne__(arg)`

`SuspensionIndicator.__repr__()`

`SuspensionIndicator.__str__()`

`tonalitytools.TonalFunction`



class `tonalitytools.TonalFunction` (*args)

New in version 2.0. Abjad model of functions in tonal harmony: I, I6, I64, V, V7, V43, V42, bII, bII6, etc., also i, i6, i64, v, v7, etc.

Value object that can not be changed after instantiation.

Read-only Properties

`TonalFunction.bass_scale_degree`

`TonalFunction.extent`

`TonalFunction.figured_bass_string`

`TonalFunction.inversion`

`TonalFunction.markup`

`TonalFunction.quality`

`TonalFunction.root_scale_degree`

`TonalFunction.scale_degree`

`TonalFunction.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`TonalFunction.suspension`

`TonalFunction.symbolic_string`

Special Methods

`TonalFunction.__eq__(arg)`

`TonalFunction.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TonalFunction.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`TonalFunction.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TonalFunction.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`TonalFunction.__ne__(arg)`

TonalFunction.__repr__()

functions

tonalitytools.analyze_chord

tonalitytools.**analyze_chord**(*expr*)

New in version 2.0. Analyze *expr* and return chord class.

```
>>> from abjad.tools import tonalitytools

>>> chord = Chord([7, 10, 12, 16], (1, 4))
>>> tonalitytools.analyze_chord(chord)
CDominantSeventhInSecondInversion
```

Return none when no tonal chord is understood.

```
>>> chord = Chord(['c', 'cs', 'd'], (1, 4))
>>> tonalitytools.analyze_chord(chord) is None
True
```

Raise tonal harmony error when chord can not analyze.

tonalitytools.analyze_incomplete_chord

tonalitytools.**analyze_incomplete_chord**(*expr*)

New in version 2.0. Analyze *expr* and return chord class based on incomplete pitches.

```
>>> from abjad.tools import tonalitytools

>>> tonalitytools.analyze_incomplete_chord(Chord([7, 11], (1, 4)))
GMajorTriadInRootPosition

>>> tonalitytools.analyze_incomplete_chord(Chord(['fs', 'g', 'b'], (1, 4)))
GMajorSeventhInSecondInversion
```

Return chord class.

tonalitytools.analyze_incomplete_tonal_function

tonalitytools.**analyze_incomplete_tonal_function**(*expr*, *key_signature*)

New in version 2.0. Analyze tonal function of *expr* according to *key_signature*:

```
>>> from abjad.tools import tonalitytools

>>> chord = Chord("<c' e'>4")
>>> key_signature = contexttools.KeySignatureMark('g', 'major')
>>> tonalitytools.analyze_incomplete_tonal_function(chord, key_signature)
IVMajorTriadInRootPosition
```

Return tonal function.

tonalitytools.analyze_tonal_function

`tonalitytools.analyze_tonal_function(expr, key_signature)`

New in version 2.0. Analyze *expr* and return tonal function according to *key_signature*.

```
>>> from abjad.tools import tonalitytools

>>> chord = Chord(['ef', 'g', 'bf'], (1, 4))
>>> key_signature = contexttools.KeySignatureMark('c', 'major')
>>> tonalitytools.analyze_tonal_function(chord, key_signature)
FlatIIIMajorTriadInRootPosition
```

Return none when no tonal function is understood.

```
>>> chord = Chord(['c', 'cs', 'd'], (1, 4))
>>> key_signature = contexttools.KeySignatureMark('c', 'major')
>>> tonalitytools.analyze_tonal_function(chord, key_signature) is None
True
```

Return tonal function or none.

tonalitytools.are_scalar_notes

`tonalitytools.are_scalar_notes(*expr)`

New in version 2.0. True when notes in *expr* are scalar.

```
>>> from abjad.tools import tonalitytools

>>> t = Staff("c'8 d'8 e'8 f'8")
>>> tonalitytools.are_scalar_notes(t[:])
True
```

Otherwise false.

```
>>> tonalitytools.are_scalar_notes(Note("c'4"), Note("c'4"))
False
```

Changed in version 2.0: renamed `tonalitytools.are_scalar()` to `tonalitytools.are_scalar_notes()`.

tonalitytools.are_stepwise_ascending_notes

`tonalitytools.are_stepwise_ascending_notes(*expr)`

New in version 2.0. True when notes in *expr* are stepwise ascending.

```
>>> from abjad.tools import tonalitytools

>>> t = Staff("c'8 d'8 e'8 f'8")
>>> tonalitytools.are_stepwise_ascending_notes(t[:])
True
```

Otherwise false.

```
>>> tonalitytools.are_stepwise_ascending_notes(Note("c'4"), Note("c'4"))
False
```

Changed in version 2.0: renamed `tonalitytools.are_stepwise_ascending()` to `tonalitytools.are_stepwise_ascending_notes()`.

tonalitytools.are_stepwise_descending_notes

`tonalitytools.are_stepwise_descending_notes(*expr)`

New in version 2.0. True when notes in *expr* are stepwise descending:

```
>>> from abjad.tools import tonalitytools

>>> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
>>> t = Staff(list(reversed(notes)))
>>> tonalitytools.are_stepwise_descending_notes(t[:])
True
```

Otherwise false:

```
>>> tonalitytools.are_stepwise_descending_notes(Note("c'4"), Note("c'4"))
False
```

Changed in version 2.0: renamed `tonalitytools.are_stepwise_descending()` to `tonalitytools.are_stepwise_descending_notes()`.

tonalitytools.are_stepwise_notes

`tonalitytools.are_stepwise_notes(*expr)`

New in version 2.0. True when notes in *expr* are stepwise.

```
>>> from abjad.tools import tonalitytools

>>> t = Staff("c'8 d'8 e'8 f'8")
>>> tonalitytools.are_stepwise_notes(t[:])
True
```

Otherwise false.

```
>>> tonalitytools.are_stepwise_notes(Note("c'4"), Note("c'4"))
False
```

Changed in version 2.0: renamed `tonalitytools.are_stepwise()` to `tonalitytools.are_stepwise_notes()`.

tonalitytools.chord_class_cardinality_to_extent

`tonalitytools.chord_class_cardinality_to_extent(cardinality)`

..versionadded:: 2.0

Change integer chord class *cardinality* to integer chord class extent:

```
>>> from abjad.tools import tonalitytools

>>> tonalitytools.chord_class_cardinality_to_extent(4)
7
```

The function above indicates that a tertian chord with 4 unique pitches qualifies as a seventh chord.

tonalitytools.chord_class_extnt_to_cardinality

`tonalitytools.chord_class_extnt_to_cardinality` (*extent*)
..versionadded:: 2.0

Change integer chord class *extent* to integer chord class cardinality:

```
>>> from abjad.tools import tonalitytools

>>> tonalitytools.chord_class_extnt_to_cardinality(7)
4
```

The call above shows that a seventh chord comprises 4 unique pitch-classes.

tonalitytools.chord_class_extnt_to_extent_name

`tonalitytools.chord_class_extnt_to_extent_name` (*extent*)
New in version 2.0. Change integer chord class *extent* to extent name.

```
>>> from abjad.tools import tonalitytools

>>> tonalitytools.chord_class_extnt_to_extent_name(7)
'seventh'
```

The call above shows that a tertian chord subtending 7 staff spaces qualifies as a seventh chord.

tonalitytools.diatonic_interval_class_segment_to_chord_quality_string

`tonalitytools.diatonic_interval_class_segment_to_chord_quality_string` (*dic_seg*)
New in version 2.0. Change diatonic interval-class segment *dic_seg* to chord quality string:

```
>>> from abjad.tools import tonalitytools

>>> dic_seg = pitchtools.InversionEquivalentDiatonicIntervalClassSegment([
...     pitchtools.InversionEquivalentDiatonicIntervalClass('major', 3),
...     pitchtools.InversionEquivalentDiatonicIntervalClass('minor', 3),])
>>> tonalitytools.diatonic_interval_class_segment_to_chord_quality_string(dic_seg)
'major'
```

Todo

Implement `diatonic_interval_class_set_to_chord_quality_string()`.

tonalitytools.is_neighbor_note

`tonalitytools.is_neighbor_note` (*note*)

New in version 2.0. True when *note* is preceded by a stepwise interval in one direction and followed by a stepwise interval in the other direction. Otherwise false.

```
>>> from abjad.tools import tonalitytools
```

```
>>> t = Staff("c'8 d'8 e'8 f'8")
>>> for note in t:
...     print '%s\t%s' % (note, tonalitytools.is_neighbor_note(note))
...
c'8      False
d'8      False
e'8      False
f'8      False
```

Return boolean.

tonalitytools.is_passing_tone

`tonalitytools.is_passing_tone` (*note*)

New in version 2.0. True when *note* is both preceeded and followed by scalewise sibling notes. Otherwise false.

```
>>> from abjad.tools import tonalitytools

>>> t = Staff("c'8 d'8 e'8 f'8")
>>> for note in t:
...     print '%s\t%s' % (note, tonalitytools.is_passing_tone(note))
...
c'8      False
d'8      True
e'8      True
f'8      False
```

Return boolean.

tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale

`tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale` (*mdi*)

New in version 2.0. True when *mdi* is unlikely melodic diatonic interval in JSB chorale.

```
>>> from abjad.tools import tonalitytools

>>> mdi = pitchtools.MelodicDiatonicInterval('major', 7)
>>> tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale(mdi)
True
```

Otherwise False.

```
>>> mdi = pitchtools.MelodicDiatonicInterval('major', 2)
>>> tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale(mdi)
False
```

Return boolean.

tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale

`tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale` (*key_signature=None*)

New in version 2.0. Construct one up-down period of scale according to *key_signature*:

```
>>> from abjad.tools import tonalitytools
```

```
>>> score = tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale(
...     contexttools.KeySignatureMark('E', 'major'))

>>> f(score)
\new Score \with {
    tempoWholesPerMinute = #(ly:make-moment 30 1)
} <<
    \new Staff {
        \key e \major
        e'8
        fs'8
        gs'8
        a'8
        b'8
        cs''8
        ds''8
        e''8
        ds''8
        cs''8
        b'8
        a'8
        gs'8
        fs'8
        e'4
    }
>>
```

Changed in version 2.0: renamed `construct.scale_period()` to `tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale()`.

tonalitytools.make_first_n_notes_in_ascending_diatonic_scale

`tonalitytools.make_first_n_notes_in_ascending_diatonic_scale` (*count*, *written_duration*=`Duration(1, 8)`, *key_signature*=`None`)

Construct *count* notes with *written_duration* according to *key_signature*:

```
>>> from abjad.tools import tonalitytools

>>> tonalitytools.make_first_n_notes_in_ascending_diatonic_scale(4)
[Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
```

Allow nonassignable *written_duration*:

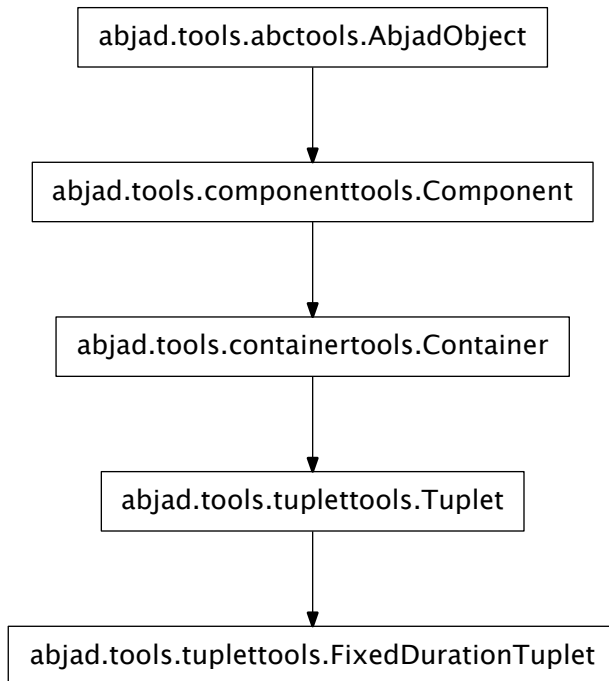
```
>>> staff = Staff(tonalitytools.make_first_n_notes_in_ascending_diatonic_scale(2, (5, 16)))
>>> f(staff)
\new Staff {
    c'4 ~
    c'16
    d'4 ~
    d'16
}
```

New in version 2.0: Optional *key_signature* keyword parameter. Changed in version 2.0: renamed `leaftools.make_first_n_notes_in_ascending_diatonic_scale()` to `tonalitytools.make_first_n_notes_in_ascending_diatonic_scale()`.

tuplettools

concrete classes

tuplettools.FixedDurationTuplet



class `tuplettools.FixedDurationTuplet` (*duration*, *music=None*, ***kwargs*)

Abjad tuplet of fixed duration and variable multiplier:

```
>>> tuplettools.FixedDurationTuplet(Fraction(2, 8), "c'8 d'8 e'8")
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
```

Return fixed-duration tuplet.

Read-only Properties

`FixedDurationTuplet.contents_duration`

Inherited from `containertools.Container`

`FixedDurationTuplet.duration_in_seconds`

Inherited from `containertools.Container`

`FixedDurationTuplet.is_augmentation`

True when multiplier is greater than 1. Otherwise false:

```
>>> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> t.is_augmentation
False
```

Return boolean.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_binary`

Read-only boolean true when multiplier numerator is power of two. Otherwise false:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
>>> tuplet.is_binary
True
```

Return boolean.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_diminution`

True when multiplier is less than 1. Otherwise false:

```
>>> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> t.is_diminution
True
```

Return boolean.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_nonbinary`

Read-only boolean true when multiplier numerator is not power of two. Otherwise false:

```
>>> tuplet = Tuplet((3, 5), "c'8 d'8 e'8 f'8 g'8")
>>> tuplet.is_nonbinary
True
```

Return boolean.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_trivial`

True when tuplet multiplier is one. Otherwise false:

```
>>> tuplet = Tuplet((1, 1), "c'8 d'8 e'8")
>>> tuplet.is_trivial
True
```

Return boolean.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.leaves`

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

`FixedDurationTuplet.lilypond_format`

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.multiplied_duration`

Read-only multiplied duration of tuplet:

```
>>> tuplet = tuplettools.FixedDurationTuplet((1, 4), "c'8 d'8 e'8")
>>> tuplet.multiplied_duration
Duration(1, 4)
```

Return duration.

`FixedDurationTuplet.music`

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

`FixedDurationTuplet.override`

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

`FixedDurationTuplet.parent`

Inherited from `componenttools.Component`

`FixedDurationTuplet.preprolated_duration`

Duration prior to prolation:

```
>>> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> t.preprolated_duration
Duration(1, 4)
```

Return duration.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.prolated_duration`

Inherited from `componenttools.Component`

`FixedDurationTuplet.prolation`

Inherited from `componenttools.Component`

`FixedDurationTuplet.ratio_string`

Read-only tuplet multiplier formatted with colon as ratio:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
>>> tuplet.ratio_string
'3:2'
```

Return string.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.set`

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

`FixedDurationTuplet`.**spanners**

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

`FixedDurationTuplet`.**start_offset**

Read-only start offset of component.

Inherited from `componenttools.Component`

`FixedDurationTuplet`.**start_offset_in_seconds**

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

`FixedDurationTuplet`.**stop_offset**

Read-only stop offset of component.

Inherited from `componenttools.Component`

`FixedDurationTuplet`.**stop_offset_in_seconds**

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

`FixedDurationTuplet`.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`FixedDurationTuplet`.**force_fraction**

Read / write boolean to force $n:m$ fraction in LilyPond format:

```
>>> tuplet = Tuplet(Fraction(2, 3), "c'8 d'8 e'8")
```

```
>>> tuplet.force_fraction is None
True
```

```
>>> f(tuplet)
\times 2/3 {
  c'8
  d'8
  e'8
}
```

```
>>> tuplet.force_fraction = True
```

```
>>> f(tuplet)
\fraction \times 2/3 {
  c'8
  d'8
  e'8
}
```

Return boolean or none.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet`.**is_invisible**

Read / write boolean to render as LilyPond `\scaledDurations` construct instead of `tuplet`:


```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
```

```
>>> f(tuplet)
\times 2/3 {
  c'8
  d'8
  e'8
}
```

```
>>> tuplet.is_invisible = True
```

```
\scaleDurations #'(2 . 3) {
  c'8
  d'8
  e'8
}
```

This has the effect of rendering no no tuplet bracket and no tuplet number while preserving the rhythmic value of the tuplet and the contents of the tuplet.

Return boolean or none.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_parallel`

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])
```

```
>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
>>> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True
```

```
>>> f(container)
<<
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
```

```
}
>>
```

Return none.

Inherited from `containertools.Container`

`FixedDurationTuplet.multiplier`

Read-only multiplier of tuplet:

```
>>> tuplet = tuplettools.FixedDurationTuplet((1, 4), "c'8 d'8 e'8")
>>> tuplet.multiplier
Fraction(2, 3)
```

Return fraction.

`FixedDurationTuplet.preferred_denominator`

New in version 2.0. Integer denominator in terms of which tuplet fraction should format:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
>>> tuplet.preferred_denominator = 4

>>> f(tuplet)
\times 4/6 {
    c'8
    d'8
    e'8
}
```

Return positive integer or none.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.target_duration`

Read / write target duration of fixed-duration tuplet:

```
>>> tuplet = tuplettools.FixedDurationTuplet((1, 4), "c'8 d'8 e'8")
>>> tuplet.target_duration
Duration(1, 4)

>>> f(tuplet)
\times 2/3 {
    c'8
    d'8
    e'8
}

>>> tuplet.target_duration = Duration(5, 8)
>>> f(tuplet)
\fraction \times 5/3 {
    c'8
    d'8
    e'8
}
```

Return duration.

Methods

`FixedDurationTuplet.append(component)`

Append *component* to container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}

```

Return none.

Inherited from `containertools.Container`

`FixedDurationTuplet.extend(expr)`

Extend *expr* against container:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```

Return none. New in version 2.3: *expr* may now be a LilyPond input string. Inherited from `containertools.Container`

`FixedDurationTuplet.index(component)`

Index *component* in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

```

```
>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

`FixedDurationTuplet.insert(i, component)`

Insert *component* in container at index *i*:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Inherited from `containertools.Container`

`FixedDurationTuplet.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

`FixedDurationTuplet.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

`FixedDurationTuplet.trim(start, stop='unused')`

Trim fixed-duration tuplet elements from *start* to *stop*:

```
>>> tuplet = tuplettools.FixedDurationTuplet(Fraction(2, 8), "c'8 d'8 e'8")
>>> tuplet
FixedDurationTuplet(1/4, [c'8, d'8, e'8])

>>> tuplet.trim(2)
>>> tuplet
FixedDurationTuplet(1/6, [c'8, d'8])
```

Preserve fixed-duration tuplet multiplier.

Adjust fixed-duration tuplet duration.

Return none.

Special Methods

`FixedDurationTuplet.__add__(arg)`

Add two tuplets of same type and with same multiplier.

Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.__contains__(expr)`

True if *expr* is in container, otherwise False.

Inherited from `containertools.Container`

`FixedDurationTuplet.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

`FixedDurationTuplet.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__getitem__(i)`
Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

`FixedDurationTuplet.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__iadd__(expr)`
`__iadd__` avoids unnecessary copying of structures.

Inherited from `containertools.Container`

`FixedDurationTuplet.__imul__(total)`
Multiply contents of container ‘total’ times. Return multiplied container.

Inherited from `containertools.Container`

`FixedDurationTuplet.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__len__()`
Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

`FixedDurationTuplet.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__mul__(n)`
Inherited from `componenttools.Component`

`FixedDurationTuplet.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`FixedDurationTuplet.__repr__()`

`FixedDurationTuplet.__rmul__(n)`

Inherited from `componenttools.Component`

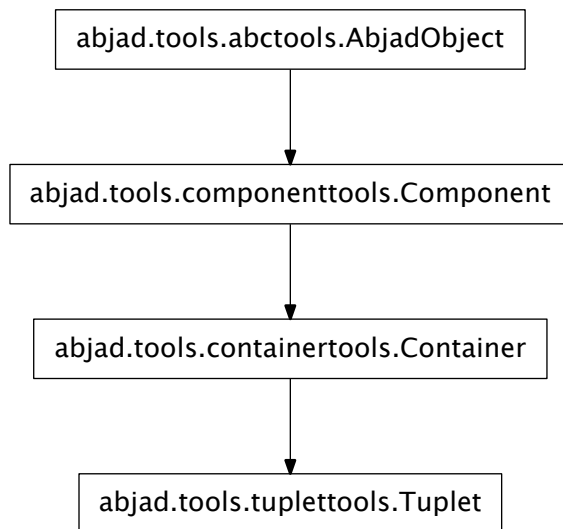
`FixedDurationTuplet.__setitem__(i, expr)`

Set ‘`expr`’ in `self` at nonnegative integer index `i`. Or, set ‘`expr`’ in `self` at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with ‘`expr`’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

`FixedDurationTuplet.__str__()`

tuplettools.Tuplet



class `tuplettools.Tuplet` (*multiplier*, *music=None*, ***kwargs*)

Abjad model of a tuplet:

```

>>> tuplet = Tuplet(Fraction(2, 3), "c'8 d'8 e'8")
>>> f(tuplet)
\times 2/3 {
  c'8
  d'8
  e'8
}
  
```

Return tuplet object.

Read-only Properties

Tuplet.contents_duration

Inherited from `containertools.Container`

Tuplet.duration_in_seconds

Inherited from `containertools.Container`

Tuplet.is_augmentation

True when multiplier is greater than 1. Otherwise false:

```
>>> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> t.is_augmentation
False
```

Return boolean.

Tuplet.is_binary

Read-only boolean true when multiplier numerator is power of two. Otherwise false:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
>>> tuplet.is_binary
True
```

Return boolean.

Tuplet.is_diminution

True when multiplier is less than 1. Otherwise false:

```
>>> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> t.is_diminution
True
```

Return boolean.

Tuplet.is_nonbinary

Read-only boolean true when multiplier numerator is not power of two. Otherwise false:

```
>>> tuplet = Tuplet((3, 5), "c'8 d'8 e'8 f'8 g'8")
>>> tuplet.is_nonbinary
True
```

Return boolean.

Tuplet.is_trivial

True when tuplet multiplier is one. Otherwise false:

```
>>> tuplet = Tuplet((1, 1), "c'8 d'8 e'8")
>>> tuplet.is_trivial
True
```

Return boolean.

Tuplet.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

Tuplet.lilypond_format

Tuplet.multiplied_duration

Read-only multiplied duration of tuplet:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
>>> tuplet.multiplied_duration
Duration(1, 4)
```

Return duration.

Tuplet.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

Tuplet.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Tuplet.parent

Inherited from `componenttools.Component`

Tuplet.preprolated_duration

Duration prior to prolation:

```
>>> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> t.preprolated_duration
Duration(1, 4)
```

Return duration.

Tuplet.prolated_duration

Inherited from `componenttools.Component`

Tuplet.prolation

Inherited from `componenttools.Component`

Tuplet.ratio_string

Read-only tuplet multiplier formatted with colon as ratio:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
>>> tuplet.ratio_string
'3:2'
```

Return string.

Tuplet.set

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Tuplet.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Tuplet.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

Tuplet.start_offset_in_seconds

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

Tuplet.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

Tuplet.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Tuplet.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

Tuplet.force_fraction

Read / write boolean to force `n:m` fraction in LilyPond format:

```
>>> tuplet = Tuplet(Fraction(2, 3), "c'8 d'8 e'8")
```

```
>>> tuplet.force_fraction is None
True
```

```
>>> f(tuplet)
\times 2/3 {
  c'8
  d'8
  e'8
}
```

```
>>> tuplet.force_fraction = True
```

```
>>> f(tuplet)
\fraction \times 2/3 {
  c'8
  d'8
  e'8
}
```

Return boolean or none.

Tuplet.is_invisible

Read / write boolean to render as LilyPond `\scaledDurations` construct instead of `tuplet`:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
```

```
>>> f(tuplet)
\times 2/3 {
  c'8
```

```

        d'8
        e'8
    }

    >>> tuplet.is_invisible = True

    \scaleDurations #'(2 . 3) {
        c'8
        d'8
        e'8
    }

```

This has the effect of rendering no no tuplet bracket and no tuplet number while preserving the rhythmic value of the tuplet and the contents of the tuplet.

Return boolean or none.

Tuplet.is_parallel

Get parallel container:

```

    >>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

    >>> f(container)
    {
      \new Voice {
        c'8
        d'8
        e'8
      }
      \new Voice {
        g4.
      }
    }

    >>> container.is_parallel
    False

```

Return boolean.

Set parallel container:

```

    >>> container.is_parallel = True

    >>> f(container)
    <<
      \new Voice {
        c'8
        d'8
        e'8
      }
      \new Voice {
        g4.
      }
    >>

```

Return none.

Inherited from `containertools.Container`

Tuplet.multiplier

Read / write tuplet multiplier:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
>>> tuplet.multiplier
Fraction(2, 3)
```

Return fraction.

`Tuplet.preferred_denominator`

New in version 2.0. Integer denominator in terms of which tuplet fraction should format:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
>>> tuplet.preferred_denominator = 4
```

```
>>> f(tuplet)
\times 4/6 {
    c'8
    d'8
    e'8
}
```

Return positive integer or none.

Methods

`Tuplet.append(component)`

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

`Tuplet.extend(expr)`

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
```

```

    e'8 ]
}

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```

Return none. New in version 2.3: `expr` may now be a LilyPond input string. Inherited from `containertools.Container`

`Tuplet.index(component)`

Index *component* in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2

```

Return nonnegative integer.

Inherited from `containertools.Container`

`Tuplet.insert(i, component)`

Insert *component* in container at index *i*:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}

```

Return none.

Inherited from `containertools.Container`

`Tuplet.pop(i=-1)`

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

`Tuplet.remove(component)`

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

`Tuplet.__add__(arg)`

Add two tuplets of same type and with same multiplier.

`Tuplet.__contains__(expr)`
 True if *expr* is in container, otherwise False.
 Inherited from `containertools.Container`

`Tuplet.__delitem__(i)`
 Find component(s) at index or slice ‘*i*’ in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).
 Inherited from `containertools.Container`

`Tuplet.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`Tuplet.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Tuplet.__getitem__(i)`
 Return component at index *i* in container. Shallow traversal of container for numeric indices only.
 Inherited from `containertools.Container`

`Tuplet.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`Tuplet.__iadd__(expr)`
`__iadd__` avoids unnecessary copying of structures.
 Inherited from `containertools.Container`

`Tuplet.__imul__(total)`
 Multiply contents of container ‘*total*’ times. Return multiplied container.
 Inherited from `containertools.Container`

`Tuplet.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Tuplet.__len__()`
 Return nonnegative integer number of components in container.
 Inherited from `containertools.Container`

`Tuplet.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`Tuplet.__mul__(n)`
 Inherited from `componenttools.Component`

`Tuplet.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abjtools.AbjadObject`

`Tuplet.__radd__(expr)`
 Extend container by contents of `expr` to the right.
 Inherited from `containertools.Container`

`Tuplet.__repr__()`

`Tuplet.__rmul__(n)`
 Inherited from `componenttools.Component`

`Tuplet.__setitem__(i, expr)`
 Set ‘`expr`’ in self at nonnegative integer index `i`. Or, set ‘`expr`’ in self at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with ‘`expr`’. Reattach spanners to new contents. This operation leaves all score trees always in tact.
 Inherited from `containertools.Container`

`Tuplet.__str__()`

functions

`tuplettools.all_are_tuplets`

`tuplettools.all_are_tuplets(expr)`
 New in version 2.6. True when `expr` is a sequence of Abjad tuplets:

```
>>> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")

>>> tuplettools.all_are_tuplets([tuplet])
True
```

True when `expr` is an empty sequence:

```
>>> tuplettools.all_are_tuplets([])
True
```

Otherwise false:

```
>>> tuplettools.all_are_tuplets('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

`tuplettools.change_augmented_tuplets_in_expr_to_diminished`

`tuplettools.change_augmented_tuplets_in_expr_to_diminished(tuplet)`
 New in version 2.0. Multiply the written duration of the leaves in `tuplet` by the least power of 2 necessary to diminished `tuplet`.

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 4), "c'8 d'8 e'8")
>>> tuplet
FixedDurationTuplet(1/2, [c'8, d'8, e'8])
>>> tuplettools.change_augmented_tuplets_in_expr_to_diminished(tuplet)
FixedDurationTuplet(1/2, [c'4, d'4, e'4])
```

Todo

make work with nested tuplets.

Changed in version 2.0: renamed `tuplettools.augmentation_to_diminution()` to `tuplettools.change_augmented_tuplets_in_expr_to_diminished()`.

`tuplettools.change_diminished_tuplets_in_expr_to_augmented`

`tuplettools.change_diminished_tuplets_in_expr_to_augmented(tuplet)`

New in version 2.0. Divide the written duration of the leaves in *tuplet* by the least power of 2 necessary to augment *tuplet*.

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> tuplet
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
>>> tuplettools.change_diminished_tuplets_in_expr_to_augmented(tuplet)
FixedDurationTuplet(1/4, [c'16, d'16, e'16])
```

Todo

make work with nested tuplets.

Changed in version 2.0: renamed `tuplettools.diminution_to_augmentation()` to `tuplettools.change_diminished_tuplets_in_expr_to_augmented()`.

`tuplettools.change_fixed_duration_tuplets_in_expr_to_tuplets`

`tuplettools.change_fixed_duration_tuplets_in_expr_to_tuplets(expr)`

New in version 2.9. Change fixed-duration tuplets in *expr* to tuplets:

```
>>> staff = Staff(2 * tuplettools.FixedDurationTuplet((2, 8), "c'8 d'8 e'8"))

>>> staff[:]
[FixedDurationTuplet(1/4, [c'8, d'8, e'8]), FixedDurationTuplet(1/4, [c'8, d'8, e'8])]

>>> tuplettools.change_fixed_duration_tuplets_in_expr_to_tuplets(staff)
[Tuplet(2/3, [c'8, d'8, e'8]), Tuplet(2/3, [c'8, d'8, e'8])]

>>> staff[:]
[Tuplet(2/3, [c'8, d'8, e'8]), Tuplet(2/3, [c'8, d'8, e'8])]
```

Return tuplets.

tuplettools.change_tuplets_in_expr_to_fixed_duration_tuplets

tuplettools.change_tuplets_in_expr_to_fixed_duration_tuplets (*expr*)

New in version 2.9. Change tuplets in *expr* to fixed-duration tuplets:

```
>>> staff = Staff(r"\times 2/3 { c'8 d'8 e'8 } \times 2/3 { c'8 d'8 e'8 }")

>>> staff[:]
[Tuplet(2/3, [c'8, d'8, e'8]), Tuplet(2/3, [c'8, d'8, e'8])]

>>> tuplettools.change_tuplets_in_expr_to_fixed_duration_tuplets(staff)
[FixedDurationTuplet(1/4, [c'8, d'8, e'8]), FixedDurationTuplet(1/4, [c'8, d'8, e'8])]

>>> staff[:]
[FixedDurationTuplet(1/4, [c'8, d'8, e'8]), FixedDurationTuplet(1/4, [c'8, d'8, e'8])]
```

Return tuplets.

tuplettools.fix_contents_of_tuplets_in_expr

tuplettools.fix_contents_of_tuplets_in_expr (*tuplet*)

Scale *tuplet* contents by power of two if tuplet multiplier less than 1/2 or greater than 2. Return tuplet.

```
>>> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'4 d'4 e'4")
>>> tuplet
FixedDurationTuplet(1/4, [c'4, d'4, e'4])
>>> tuplettools.fix_contents_of_tuplets_in_expr(tuplet)
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
```

Changed in version 2.0: renamed `tuplettools.contents_fix()` to `tuplettools.fix_contents_of_tuplets_in_expr()`.

tuplettools.fuse_tuplets

tuplettools.fuse_tuplets (*tuplets*)

Fuse parent-contiguous *tuplets*:

```
>>> t1 = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
>>> beamtools.BeamSpanner(t1[:])
BeamSpanner(c'8, d'8, e'8)
>>> t2 = tuplettools.FixedDurationTuplet(Duration(2, 16), "c'16 d'16 e'16")
>>> spannertools.SlurSpanner(t2[:])
SlurSpanner(c'16, d'16, e'16)
>>> staff = Staff([t1, t2])
>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8 [
      d'8
      e'8 ]
  }
  \times 2/3 {
    c'16 (
      d'16
      e'16 )
  }
}
```

```

    }
}

>>> tuplettools.fuse_tuplets(staff[:])
FixedDurationTuplet(3/8, [c'8, d'8, e'8, c'16, d'16, e'16])

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8 [
      d'8
      e'8 ]
    c'16 (
      d'16
      e'16 )
  }
}

```

Return new tuplet.

Fuse zero or more parent-contiguous *tuplets*.

Allow in-score *tuplets*.

Allow outside-of-score *tuplets*.

All *tuplets* must carry the same multiplier.

All *tuplets* must be of the same type. Changed in version 2.0: renamed `fuse.tuplets_by_reference()` to `tuplettools.fuse_tuplets()`.

tuplettools.get_first_tuplet_in_improper_parentage_of_component

`tuplettools.get_first_tuplet_in_improper_parentage_of_component` (*component*)

New in version 2.0. Get first tuplet in improper parentage of *component*:

```

>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> Tuplet(Fraction(2, 3), staff[:3])
Tuplet(2/3, [c'8, d'8, e'8])

>>> f(staff)
\new Staff {
  \times 2/3 {
    c'8
    d'8
    e'8
  }
  f'8
}

>>> tuplettools.get_first_tuplet_in_improper_parentage_of_component(staff.leaves[1])
Tuplet(2/3, [c'8, d'8, e'8])

```

Return tuplet or none.

`tuplettools.get_first_tuplet_in_proper_parentage_of_component`

```
tuplettools.get_first_tuplet_in_proper_parentage_of_component(component)
```

New in version 2.0. Get first tuple in proper parentage of *component*:

```
>>> staff = Staff("c'8 d'8 e'8 f'8")
>>> Tuplet(Fraction(2, 3), staff[:3])
Tuplet(2/3, [c'8, d'8, e'8])

>>> f(staff)
\new Staff {
    \times 2/3 {
        c'8
        d'8
        e'8
    }
    f'8
}

>>> tuplettools.get_first_tuplet_in_proper_parentage_of_component(staff.leaves[1])
Tuplet(2/3, [c'8, d'8, e'8])
```

Return tuple or none.

tuplettools.leaf to tuplet with n notes of equal written duration

[illegible]

Change *leaf* to tuplet n notes of equal written duration.

Example 1. Change leaf to augmented tuple:

```
>>> for n in range(1, 11):
...     note = Note(0, (3, 16))
...     tuplet = tuplettools.leaf_to_tuplet_with_n_notes_of_equal_written_duration(
...         note, n, diminution=False)
...     print tuplet
...
{@ 1:1 c'8. @}
{@ 1:1 c'16., c'16. @}
{@ 1:1 c'16, c'16, c'16 @}
{@ 1:1 c'32., c'32., c'32., c'32. @}
{@ 5:8 c'64., c'64., c'64., c'64., c'64. @}
{@ 1:1 c'32, c'32, c'32, c'32, c'32 @}
{@ 7:8 c'64., c'64., c'64., c'64., c'64., c'64., c'64. @}
{@ 1:1 c'64., c'64., c'64., c'64., c'64., c'64., c'64., c'64. @}
{@ 3:4 c'64, c'64, c'64, c'64, c'64, c'64, c'64, c'64 @}
{@ 5:8 c'128., c'128., c'128., c'128., c'128., c'128., c'128., c'128., c'128., c'128. @}
```

Example 2. Change to leaf diminished tuplet:

```
>>> for n in range(1, 11):
...     note = Note(0, (3, 16))
...     tuplet = tuplettools.leaf_to_tuplet_with_n_notes_of_equal_written_duration(
...         note, n, diminution=True)
...     print tuplet
... 
```

```
{@ 1:1 c'8. @}
{@ 1:1 c'16., c'16. @}
{@ 1:1 c'16, c'16, c'16 @}
{@ 1:1 c'32., c'32., c'32., c'32. @}
{@ 5:4 c'32., c'32., c'32., c'32., c'32. @}
{@ 1:1 c'32, c'32, c'32, c'32, c'32, c'32 @}
{@ 7:4 c'32., c'32., c'32., c'32., c'32., c'32., c'32. @}
{@ 1:1 c'64., c'64., c'64., c'64., c'64., c'64., c'64., c'64. @}
{@ 3:2 c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32 @}
{@ 5:4 c'64., c'64., c'64., c'64., c'64., c'64., c'64., c'64., c'64., c'64. @}
```

Return fixed-duration tuplet.

tuplettools.leaf_to_tuplet_with_proportions

`tuplettools.leaf_to_tuplet_with_proportions(leaf, proportions, diminution=True)`

Change *leaf* to tuplet with *proportions*.

Example 1. Change *leaf* to augmented tuplet with *proportions*:

```
>>> note = Note(0, (3, 16))

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1], diminution=False)
{@ 1:1 c'8. @}

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2], diminution=False)
{@ 1:1 c'16, c'8 @}

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2, 2], diminution=False)
{@ 5:8 c'64., c'32., c'32. @}

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2, 2, 3], diminution=False)
{@ 2:3 c'64, c'32, c'32, c'32. @}

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2, 2, 3, 3], diminution=False)
{@ 11:12 c'64, c'32, c'32, c'32., c'32. @}

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2, 2, 3, 3, 4], diminution=False)
{@ 5:8 c'128, c'64, c'64, c'64., c'64., c'32 @}
```

Example 2. Change *leaf* to diminished tuplet:

```
>>> note = Note(0, (3, 16))

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1], diminution=True)
{@ 1:1 c'8. @}

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2], diminution=True)
{@ 1:1 c'16, c'8 @}
```

```
>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2, 2], diminution=True)
{@ 5:4 c'32., c'16., c'16. @}

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2, 2, 3], diminution=True)
{@ 4:3 c'32, c'16, c'16, c'16. @}

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2, 2, 3, 3], diminution=True)
{@ 11:6 c'32, c'16, c'16, c'16., c'16. @}

>>> print tuplettools.leaf_to_tuplet_with_proportions(
...     note, [1, 2, 2, 3, 3, 4], diminution=True)
{@ 5:4 c'64, c'32, c'32, c'32., c'32., c'16 @}

```

Return fixed-duration tuplet.

tuplettools.make_tuplet_from_duration_and_proportions

```
tuplettools.make_tuplet_from_duration_and_proportions(duration, proportions,
                                                    avoid_dots=True,
                                                    big_endian=True,
                                                    is_diminution=True)
```

New in version 2.10. Make tuplet from *duration* and *proportions*.

Example set 1. Make augmented tuplet from *duration* and *proportions* and avoid dots.

Return tupletted leaves strictly without dots when all *proportions* equal 1:

```
>>> print tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [1, 1, 1, -1, -1], avoid_dots=True, is_diminution=False)
{@ 5:6 c'32, c'32, c'32, r32, r32 @}

```

Allow tupletted leaves to return with dots when some *proportions* do not equal 1:

```
>>> print tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [1, -2, -2, 3, 3], avoid_dots=True, is_diminution=False)
{@ 11:12 c'64, r32, r32, c'32., c'32. @}

```

Interpret nonassignable *proportions* according to *big_endian*:

```
>>> print tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [5, -1, 5], avoid_dots=True, big_endian=False, is_diminution=False)
{@ 11:12 c'64, c'16, r64, c'64, c'16 @}

```

Example set 2. Make augmented tuplet from *duration* and *proportions* and encourage dots:

```
>>> tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [1, 1, 1, -1, -1], avoid_dots=False, is_diminution=False)
FixedDurationTuplet(3/16, [c'64., c'64., c'64., r64., r64.])

```

Interpret nonassignable *proportions* according to *big_endian*:

```
>>> tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [5, -1, 5], avoid_dots=False, big_endian=False, is_diminution=False)
FixedDurationTuplet(3/16, [c'32..., r128., c'32...])

```

Example set 3. Make diminished tuplet from *duration* and nonzero integer *proportions*.

Return tupletted leaves strictly without dots when all *proportions* equal 1:

```
>>> print tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [1, 1, 1, -1, -1], avoid_dots=True, is_diminution=True)
{@ 5:3 c'16, c'16, c'16, r16, r16 @}
```

Allow tupletted leaves to return with dots when some *proportions* do not equal 1:

```
>>> print tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [1, -2, -2, 3, 3], avoid_dots=True, is_diminution=True)
{@ 11:6 c'32, r16, r16, c'16., c'16. @}
```

Interpret nonassignable *proportions* according to *big_endian*:

```
>>> print tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [5, -1, 5], avoid_dots=True, big_endian=False, is_diminution=True)
{@ 11:6 c'32, c'8, r32, c'32, c'8 @}
```

Example set 4. Make diminished tuplet from *duration* and *proportions* and encourage dots:

```
>>> tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [1, 1, 1, -1, -1], avoid_dots=False, is_diminution=True)
FixedDurationTuplet(3/16, [c'32., c'32., c'32., r32., r32.])
```

Interpret nonassignable *proportions* according to *direction*:

```
>>> tuplettools.make_tuplet_from_duration_and_proportions(
... Fraction(3, 16), [5, -1, 5], avoid_dots=False, big_endian=False, is_diminution=True)
FixedDurationTuplet(3/16, [c'16..., r64., c'16...])
```

Reduce *proportions* relative to each other.

Interpret negative *proportions* as rests.

Return fixed-duration tuplet.

tuplettools.make_tuplet_from_proportions_and_pair

tuplettools.**make_tuplet_from_proportions_and_pair**(*l*, (*n*, *d*))

Divide (*n*, *d*) according to *l*.

Where no prolotion is necessary, return container.

```
>>> tuplettools.make_tuplet_from_proportions_and_pair([1], (7, 16))
{c'4..}
```

Where prolotion is necessary, return fixed-duration tuplet.

```
>>> tuplettools.make_tuplet_from_proportions_and_pair([1, 2], (7, 16))
FixedDurationTuplet(7/16, [c'8, c'4])
```

```
>>> tuplettools.make_tuplet_from_proportions_and_pair([1, 2, 4], (7, 16))
FixedDurationTuplet(7/16, [c'16, c'8, c'4])
```

```
>>> tuplettools.make_tuplet_from_proportions_and_pair([1, 2, 4, 1], (7, 16))
FixedDurationTuplet(7/16, [c'16, c'8, c'4, c'16])
```

```
>>> tuplettools.make_tuplet_from_proportions_and_pair([1, 2, 4, 1, 2], (7, 16))
FixedDurationTuplet(7/16, [c'16, c'8, c'4, c'16, c'8])

>>> tuplettools.make_tuplet_from_proportions_and_pair([1, 2, 4, 1, 2, 4], (7, 16))
FixedDurationTuplet(7/16, [c'16, c'8, c'4, c'16, c'8, c'4])
```

Note: function accepts a pair rather than a rational.

Note: function interprets *d* as tuplet denominator.

Changed in version 2.0: renamed `divide.pair()` to `tuplettools.make_tuplet_from_proportions_and_pair`

tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet

`tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet` (*tuplet*)
 Scale tuplet contents and then bequeath in-score position of tuplet to contents.

Return orphaned tuplet emptied of all contents.

```
>>> t = Staff(tuplettools.FixedDurationTuplet(Duration(3, 8), "c'8 d'8") * 2)
>>> beamtools.BeamSpanner(t.leaves)
BeamSpanner(c'8, d'8, c'8, d'8)

>>> f(t)
\new Staff {
  \fraction \times 3/2 {
    c'8 [
      d'8
    ]
  }
  \fraction \times 3/2 {
    c'8
    d'8 ]
  }
}

>>> tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet(t[0])
FixedDurationTuplet(3/8, [])

>>> f(t)
\new Staff {
  c'8. [
    d'8.
  ]
  \fraction \times 3/2 {
    c'8
    d'8 ]
  }
}
```

Changed in version 2.0: renamed `tuplettools.subsume()` to `tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet()`.

`tuplettools.remove_trivial_tuplets_in_expr`

`tuplettools.remove_trivial_tuplets_in_expr` (*expr*)

Remove trivial tuplets in *expr*:

```
>>> t = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8 e'8")
>>> u = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8")
>>> s = Staff([t, u])
>>> len(s)
2

>>> s[0]
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
>>> s[1]
FixedDurationTuplet(1/4, [c'8, d'8])

>>> tuplettools.remove_trivial_tuplets_in_expr(s)
>>> len(s)
3
>>> s[0]
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
>>> s[1]
Note("c'8")
>>> s[2]
Note("d'8")

>>> f(s)
\new Staff {
  \times 2/3 {
    c'8
    d'8
    e'8
  }
  c'8
  d'8
}
```

Replace trivial tuplets with plain leaves.

Return `none`. Changed in version 2.0: renamed `tuplettools.slip_trivial()` to `tuplettools.remove_trivial_tuplets_in_expr()`.

`tuplettools.scale_contents_of_tuplets_in_expr_by_multiplier`

`tuplettools.scale_contents_of_tuplets_in_expr_by_multiplier` (*tuplet*, *multiplier*)

Scale fixed-duration tuplet by multiplier. Preserve tuplet multiplier. Return tuplet.

`tuplettools.set_denominator_of_tuplets_in_expr_to_at_least`

`tuplettools.set_denominator_of_tuplets_in_expr_to_at_least` (*expr*, *n*)

New in version 2.0. Set denominator of tuplets in *expr* to at least *n*:

```
>>> tuplet = Tuplet(Fraction(3, 5), "c'4 d'8 e'8 f'4 g'2")
```

```
>>> f(tuplet)
\fraction \times 3/5 {
    c' 4
    d' 8
    e' 8
    f' 4
    g' 2
}

>>> tuplettools.set_denominator_of_tuplets_in_expr_to_at_least(tuplet, 8)

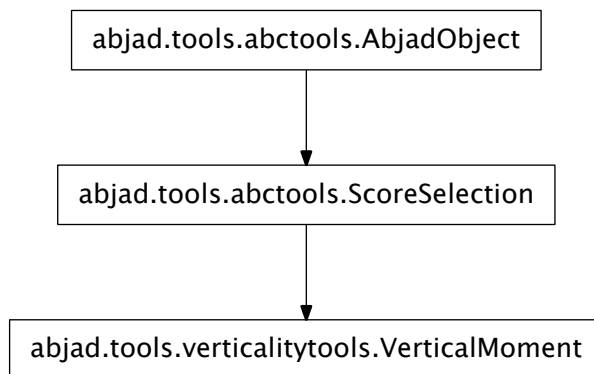
>>> f(tuplet)
\fraction \times 6/10 {
    c' 4
    d' 8
    e' 8
    f' 4
    g' 2
}
```

Return none.

verticalitytools

concrete classes

verticalitytools.VerticalMoment



class `verticalitytools.VerticalMoment` (*prolated_offset, governors, components*)
 Everything happening at a single moment in musical time:

```
>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> piano_staff = scoretools.PianoStaff([])
```

```

>>> piano_staff.append(Staff("c'4 e'4 d'4 f'4"))
>>> piano_staff.append(Staff(r"""\clef "bass" g2 f2"""))
>>> score.append(piano_staff)

f(score)
\new Score <<
  \new PianoStaff <<
    \new Staff {
      c'4
      e'4
      d'4
      f'4
    }
    \new Staff {
      \clef "bass"
      g2
      f2
    }
  >>
>>

>>> for x in verticalitytools.iterate_vertical_moments_in_expr(score):
...     x
...
VerticalMoment(0, <<2>>)
VerticalMoment(1/4, <<2>>)
VerticalMoment(1/2, <<2>>)
VerticalMoment(3/4, <<2>>)

```

Create vertical moments with the getters and iterators implemented in the `verticalitytools` module.

Vertical moments are immutable.

Read-only Properties

`VerticalMoment.attack_count`

Positive integer number of pitch carriers starting at vertical moment.

`VerticalMoment.components`

Read-only tuple of zero or more components happening at vertical moment.

It is always the case that `self.components = self.overlap_components + self.start_components`.

`VerticalMoment.governors`

Read-only tuple of one or more containers in which vertical moment is evaluated.

`VerticalMoment.leaves`

Read-only tuple of zero or more leaves at vertical moment.

`VerticalMoment.measures`

Read-only tuple of zero or more measures at vertical moment.

`VerticalMoment.music`

Read-only tuple of components in selection.

Inherited from `abctools.ScoreSelection`

`VerticalMoment.next_vertical_moment`

Read-only reference to next vertical moment forward in time.

`VerticalMoment.notes`

Read-only tuple of zero or more notes at vertical moment.

`VerticalMoment.overlap_components`

Read-only tuple of components in vertical moment starting before vertical moment, ordered by score index.

`VerticalMoment.overlap_leaves`

Read-only tuple of leaves in vertical moment starting before vertical moment, ordered by score index.

`VerticalMoment.overlap_measures`

Read-only tuple of measures in vertical moment starting before vertical moment, ordered by score index.

`VerticalMoment.overlap_notes`

Read-only tuple of notes in vertical moment starting before vertical moment, ordered by score index.

`VerticalMoment.prev_vertical_moment`

Read-only reference to prev vertical moment backward in time.

`VerticalMoment.prolated_offset`

Read-only rational-valued score offset at which vertical moment is evaluated.

`VerticalMoment.start_components`

Read-only tuple of components in vertical moment starting with at vertical moment, ordered by score index.

`VerticalMoment.start_leaves`

Read-only tuple of leaves in vertical moment starting with vertical moment, ordered by score index.

`VerticalMoment.start_notes`

Read-only tuple of notes in vertical moment starting with vertical moment, ordered by score index.

`VerticalMoment.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`VerticalMoment.__contains__(expr)`

Inherited from `abctools.ScoreSelection`

`VerticalMoment.__eq__(expr)`

`VerticalMoment.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`VerticalMoment.__getitem__(expr)`

Inherited from `abctools.ScoreSelection`

`VerticalMoment.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`VerticalMoment.__hash__()`

`VerticalMoment.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`VerticalMoment.__len__()`

`VerticalMoment.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`VerticalMoment.__ne__(expr)`

`VerticalMoment.__repr__()`

functions

`verticalitytools.get_vertical_moment_at_offset_in_expr`

`verticalitytools.get_vertical_moment_at_offset_in_expr(expr, prolated_offset)`

New in version 2.0. Get vertical moment at *prolated_offset* in *expr*:

```
>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff(r"\times 4/3 { d''8 c''8 b'8 }")
>>> score.append(staff)

>>> piano_staff = scoretools.PianoStaff([])
>>> piano_staff.append(Staff("a'4 g'4"))
>>> piano_staff.append(Staff(r"""\clef "bass" f'8 e'8 d'8 c'8"""))
>>> score.append(piano_staff)

>>> f(score)
\new Score <<
  \new Staff {
    \fraction \times 4/3 {
      d''8
      c''8
      b'8
    }
  }
  \new PianoStaff <<
    \new Staff {
      a'4
      g'4
    }
    \new Staff {
      \clef "bass"
      f'8
      e'8
      d'8
      c'8
    }
  }
>>
>>
```

```
>>> args = (piano_staff, durationtools.Offset(1, 8))

>>> verticalitytools.get_vertical_moment_at_offset_in_expr(*args)
VerticalMoment(1/8, <<2>>)

>>> vertical_moment = _
>>> vertical_moment.leaves
(Note("a'4"), Note("e'8"))
```

Return vertical moment.

verticalitytools.get_vertical_moment_starting_with_component

verticalitytools.get_vertical_moment_starting_with_component (*component*, *governor=None*)

New in version 2.0. Get vertical moment starting with *component*:

```
>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff(r"\times 4/3 { d''8 c''8 b'8 }")
>>> score.append(staff)

>>> piano_staff = scoretools.PianoStaff([])
>>> piano_staff.append(Staff("a'4 g'4"))
>>> piano_staff.append(Staff(r"""\clef "bass" f'8 e'8 d'8 c'8"""))
>>> score.append(piano_staff)

>>> f(score)
\new Score <<
  \new Staff {
    \fraction \times 4/3 {
      d''8
      c''8
      b'8
    }
  }
  \new PianoStaff <<
    \new Staff {
      a'4
      g'4
    }
    \new Staff {
      \clef "bass"
      f'8
      e'8
      d'8
      c'8
    }
  }
>>

>>> leaf = piano_staff[1][1]

>>> verticalitytools.get_vertical_moment_starting_with_component(leaf)
VerticalMoment(1/8, <<3>>)
```

Get vertical moment starting with *component* in *governor* when *governor* is not none:

```
>>> verticalitytools.get_vertical_moment_starting_with_component(leaf,
... governor=piano_staff)
VerticalMoment(1/8, <<2>>)
```

Return vertical moment.

verticalitytools.iterate_vertical_moments_in_expr

`verticalitytools.iterate_vertical_moments_in_expr(expr, reverse=False)`

New in version 2.10. Iterate vertical moments forward in *expr*:

```
>>> from abjad.tools import verticalitytools

>>> score = Score([])
>>> staff = Staff(r"\times 4/3 { d''8 c''8 b'8 }")
>>> score.append(staff)

>>> piano_staff = scoretools.PianoStaff([])
>>> piano_staff.append(Staff("a'4 g'4"))
>>> piano_staff.append(Staff(r"""\clef "bass" f'8 e'8 d'8 c'8"""))
>>> score.append(piano_staff)

>>> f(score)
\new Score <<
  \new Staff {
    \fraction \times 4/3 {
      d''8
      c''8
      b'8
    }
  }
  \new PianoStaff <<
    \new Staff {
      a'4
      g'4
    }
    \new Staff {
      \clef "bass"
      f'8
      e'8
      d'8
      c'8
    }
  }
>>

>>> for x in verticalitytools.iterate_vertical_moments_in_expr(score):
...     x.leaves
...
(Note("d''8"), Note("a'4"), Note("f'8"))
(Note("d''8"), Note("a'4"), Note("e'8"))
(Note("c''8"), Note("a'4"), Note("e'8"))
(Note("c''8"), Note("g'4"), Note("d'8"))
(Note("b'8"), Note("g'4"), Note("d'8"))
(Note("b'8"), Note("g'4"), Note("c'8"))
```

```
>>> for x in verticalitytools.iterate_vertical_moments_in_expr(piano_staff):
...     x.leaves
...
(Note("a'4"), Note("f'8"))
(Note("a'4"), Note("e'8"))
(Note("g'4"), Note("d'8"))
(Note("g'4"), Note("c'8"))
```

Iterate vertical moments backward in *expr*:

```
::
```

```
>>> for x in verticalitytools.iterate_vertical_moments_in_expr(score, reverse=True):
...     x.leaves
...
(Note("b'8"), Note("g'4"), Note("c'8"))
(Note("b'8"), Note("g'4"), Note("d'8"))
(Note("c''8"), Note("g'4"), Note("d'8"))
(Note("c''8"), Note("a'4"), Note("e'8"))
(Note("d''8"), Note("a'4"), Note("e'8"))
(Note("d''8"), Note("a'4"), Note("f'8"))

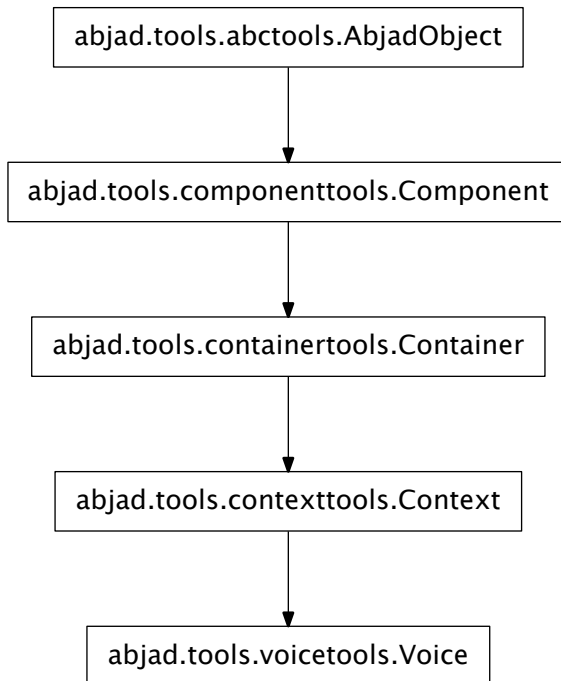
>>> for x in verticalitytools.iterate_vertical_moments_in_expr(piano_staff, reverse=True):
...     x.leaves
...
(Note("g'4"), Note("c'8"))
(Note("g'4"), Note("d'8"))
(Note("a'4"), Note("e'8"))
(Note("a'4"), Note("f'8"))
```

Return generator.

voicetools

concrete classes

voicetools.Voice



class `voicetools.Voice` (*music=None, context_name='Voice', name=None*)

Abjad model of a voice:

```

>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> f(voice)
\new Voice {
  c'8
  d'8
  e'8
  f'8
}

```

Return voice object.

Read-only Properties

`Voice.contents_duration`

Inherited from `containertools.Container`

`Voice.duration_in_seconds`

Inherited from `containertools.Container`

Voice.engraver_consists

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_consists.append('Horizontal_bracket_engraver')
>>> f(staff)
\new Staff \with {
    \consists Horizontal_bracket_engraver
} {
}
```

Inherited from `contexttools.Context`

Voice.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
>>> staff = Staff([])
>>> staff.engraver_removals.append('Time_signature_engraver')
>>> f(staff)
\new Staff \with {
    \remove Time_signature_engraver
} {
}
```

Inherited from `contexttools.Context`

Voice.is_semantic

Inherited from `contexttools.Context`

Voice.leaves

Read-only tuple of leaves in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Inherited from `containertools.Container`

Voice.lilypond_format

Inherited from `contexttools.Context`

Voice.music

Read-only tuple of components in container:

```
>>> container = Container("c'8 d'8 e'8")

>>> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Inherited from `containertools.Container`

Voice.override

Read-only reference to LilyPond grob override component plug-in.

Inherited from `componenttools.Component`

Voice.parent

Inherited from `componenttools.Component`

Voice.preprolated_duration

Inherited from `containertools.Container`

Voice.prolated_duration

Inherited from `componenttools.Component`

Voice.prolation

Inherited from `componenttools.Component`

Voice.set

Read-only reference LilyPond context setting component plug-in.

Inherited from `componenttools.Component`

Voice.spanners

Read-only reference to unordered set of spanners attached to component.

Inherited from `componenttools.Component`

Voice.start_offset

Read-only start offset of component.

Inherited from `componenttools.Component`

Voice.start_offset_in_seconds

Read-only start offset of component in seconds.

Inherited from `componenttools.Component`

Voice.stop_offset

Read-only stop offset of component.

Inherited from `componenttools.Component`

Voice.stop_offset_in_seconds

Read-only stop offset of component in seconds.

Inherited from `componenttools.Component`

Voice.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties**Voice.context_name**

Read / write name of context as a string.

Inherited from `contexttools.Context`

Voice.is_nonsemantic

Set indicator of nonsemantic voice:

```
>>> measures = measuretools.make_measures_with_full_measure_spacer_skips(  
...     [(1, 8), (5, 16), (5, 16)])  
>>> voice = Voice(measures)  
>>> voice.name = 'HiddenTimeSignatureVoice'
```

```
>>> voice.is_nonsemantic = True

>>> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    s1 * 5/16
  }
}
```

```
>>> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
>>> voice = Voice([])

>>> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Inherited from `contexttools.Context`

Voice.is_parallel

Get parallel container:

```
>>> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

>>> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}

>>> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
>>> container.is_parallel = True

>>> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Inherited from `containertools.Container`

Voice.name

Read-write name of context. Must be string or none.

Inherited from `contexttools.Context`

Methods

Voice.append(*component*)

Append *component* to container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.append(Note("f'8"))

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Inherited from `containertools.Container`

Voice.extend(*expr*)

Extend *expr* against container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
```

```

        d'8
        e'8 ]
    }

>>> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: `expr` may now be a LilyPond input string. Inherited from `containertools.Container`

Voice.index(*component*)

Index *component* in container:

```

>>> container = Container("c'8 d'8 e'8")

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.index(note)
2
```

Return nonnegative integer.

Inherited from `containertools.Container`

Voice.insert(*i*, *component*)

Insert *component* in container at index *i*:

```

>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.insert(1, Note("cs'8"))

>>> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Voice.**pop** (*i=-1*)

Pop component at index *i* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> container.pop(-1)
Note("e'8")

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Inherited from `containertools.Container`

Voice.**remove** (*component*)

Remove *component* from container:

```
>>> container = Container("c'8 d'8 e'8")
>>> beam = beamtools.BeamSpanner(container.music)

>>> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

>>> note = container[-1]
>>> note
Note("e'8")

>>> container.remove(note)

>>> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Inherited from `containertools.Container`

Special Methods

Voice.**__add__** (*expr*)

Concatenate containers self and *expr*. The operation `c = a + b` returns a new Container *c* with the content of both

a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Inherited from `containertools.Container`

Voice. **__contains__** (*expr*)

True if *expr* is in container, otherwise False.

Inherited from `containertools.Container`

Voice. **__delitem__** (*i*)

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Inherited from `containertools.Container`

Voice. **__eq__** (*arg*)

True when `id(self) equals id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

Voice. **__ge__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Voice. **__getitem__** (*i*)

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Inherited from `containertools.Container`

Voice. **__gt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Voice. **__iadd__** (*expr*)

__iadd__ avoids unnecessary copying of structures.

Inherited from `containertools.Container`

Voice. **__imul__** (*total*)

Multiply contents of container 'total' times. Return multiplied container.

Inherited from `containertools.Container`

Voice. **__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Voice. **__len__** ()

Return nonnegative integer number of components in container.

Inherited from `containertools.Container`

Voice. **__lt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Voice.__mul__(n)`

Inherited from `componenttools.Component`

`Voice.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Voice.__radd__(expr)`

Extend container by contents of `expr` to the right.

Inherited from `containertools.Container`

`Voice.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Inherited from `contexttools.Context`

`Voice.__rmul__(n)`

Inherited from `componenttools.Component`

`Voice.__setitem__(i, expr)`

Set ‘`expr`’ in `self` at nonnegative integer index `i`. Or, set ‘`expr`’ in `self` at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with ‘`expr`’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Inherited from `containertools.Container`

functions

`voicetools.all_are_voices`

`voicetools.all_are_voices(expr)`

New in version 2.6. True when `expr` is a sequence of Abjad voices:

```
>>> voice = Voice("c'4 d'4 e'4 f'4")
```

```
>>> voicetools.all_are_voices([voice])
True
```

True when `expr` is an empty sequence:

```
>>> voicetools.all_are_voices([])
True
```

Otherwise false:

```
>>> voicetools.all_are_voices('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

voicetools.get_first_voice_in_improper_parentage_of_component

`voicetools.get_first_voice_in_improper_parentage_of_component` (*component*)

New in version 2.0. Get first voice in improper parentage of *component*:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> staff = Staff([voice])

>>> f(staff)
\new Staff {
  \new Voice {
    c'8
    d'8
    e'8
    f'8
  }
}

>>> voicetools.get_first_voice_in_improper_parentage_of_component(staff.leaves[0])
Voice{4}
```

Return voice or none.

voicetools.get_first_voice_in_proper_parentage_of_component

`voicetools.get_first_voice_in_proper_parentage_of_component` (*component*)

New in version 2.0. Get first voice in proper parentage of *component*:

```
>>> voice = Voice("c'8 d'8 e'8 f'8")
>>> staff = Staff([voice])

>>> f(staff)
\new Staff {
  \new Voice {
    c'8
    d'8
    e'8
    f'8
  }
}

>>> voicetools.get_first_voice_in_proper_parentage_of_component(staff.leaves[0])
Voice{4}
```

Return voice or none.

8.1.2 Internal packages

abctools

abstract classes

abctools.AbjadObject

abjad.tools.abctools.AbjadObject

class abctools.**AbjadObject** (*args, **kwargs)

New in version 2.0. Abstract base class from which all custom classes should inherit.

Abjad objects raise exceptions on `__gt__`, `__ge__`, `__lt__`, `__le__`.

Abjad objects compare equal only with equal object IDs.

Authors of custom classes should override these behaviors as required.

Read-only Properties

AbjadObject.**storage_format**

Storage format of Abjad object.

Return string.

Special Methods

AbjadObject.**__eq__**(arg)

True when `id(self)` equals `id(arg)`.

Return boolean.

AbjadObject.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

AbjadObject.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

AbjadObject.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

AbjadObject.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

`AbjadObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

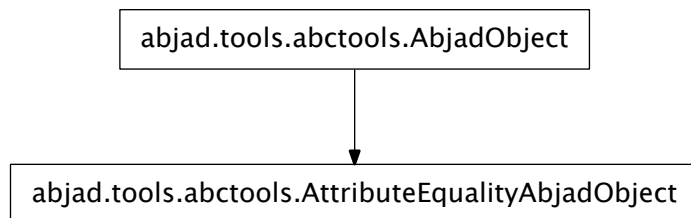
Return boolean.

`AbjadObject.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

abctools.AttributeEqualityAbjadObject



class `abctools.AttributeEqualityAbjadObject(*args, **kwargs)`

New in version 2.0. Abstract base class to confer nonsorting attribute-equality to any custom class.

Nonsorting objects raise exceptions on `__gt__`, `__ge__`, `__lt__`, `__le__`.

Attribute-equality objects compare equal only with equal comparison attributes.

Read-only Properties

`AttributeEqualityAbjadObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`AttributeEqualityAbjadObject.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

`AttributeEqualityAbjadObject.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AttributeEqualityAbjadObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`AttributeEqualityAbjadObject.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AttributeEqualityAbjadObject.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

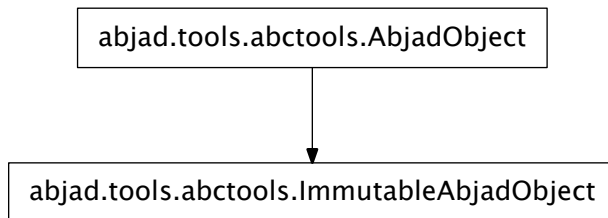
`AttributeEqualityAbjadObject.__ne__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

`AttributeEqualityAbjadObject.__repr__()`
 Interpreter representation of object defined equal to class name and format string.

Return string.

abctools.ImmutableAbjadObject



class `abctools.ImmutableAbjadObject` (*args, **kwargs)

New in version 2.8. Abstract base class from which all custom classes which also subclass immutable builtin classes, such as tuple and frozenset, should inherit.

Read-only Properties

`ImmutableAbjadObject.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ImmutableAbjadObject.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

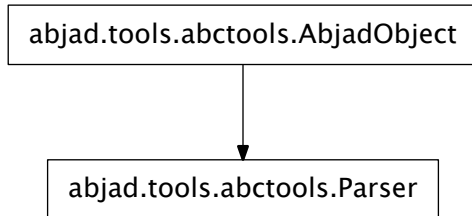
`ImmutableAbjadObject.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.
Inherited from `abctools.AbjadObject`

abctools.Parser

class `abctools.Parser` (*debug=False*)

Abstract base class for Abjad parsers.

Rules objects for lexing and parsing must be defined by overriding the abstract properties *lexer_rules_object* and *parser_rules_object*.

For most parsers these properties should simply return *self*.

Read-only Properties

`Parser.debug`

True if the parser runs in debugging mode.

`Parser.lexer`

The parser's PLY Lexer instance.

`Parser.lexer_rules_object`

The object containing the parser's lexical rule definitions.

`Parser.logger`

The parser's Logger instance.

`Parser.logger_path`

The output path for the parser's logfile.

`Parser.output_path`

The output path for files associated with the parser.

`Parser.parser`

The parser's PLY LRPParser instance.

`Parser.parser_rules_object`

The object containing the parser's syntactical rule definitions.

`Parser.pickle_path`

The output path for the parser's pickled parsing tables.

`Parser.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`Parser.tokenize(input_string)`
Tokenize *input string* and print results.

Special Methods

`Parser.__call__(input_string)`
Parse *input_string* and return result.

`Parser.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Parser.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Parser.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Parser.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Parser.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Parser.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

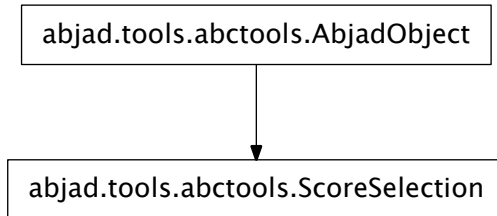
Return boolean.

Inherited from `abctools.AbjadObject`

`Parser.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

abctools.ScoreSelection

class `abctools.ScoreSelection` (*music*)

New in version 2.9. Abstract base class from which selection classes inherit.

Score selections are immutable and never change after instantiation.

Read-only Properties

`ScoreSelection.music`

Read-only tuple of components in selection.

`ScoreSelection.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ScoreSelection.__contains__` (*expr*)

`ScoreSelection.__eq__` (*expr*)

`ScoreSelection.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ScoreSelection.__getitem__` (*expr*)

`ScoreSelection.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ScoreSelection.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ScoreSelection.__len__` ()

`ScoreSelection.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ScoreSelection.__ne__(expr)`

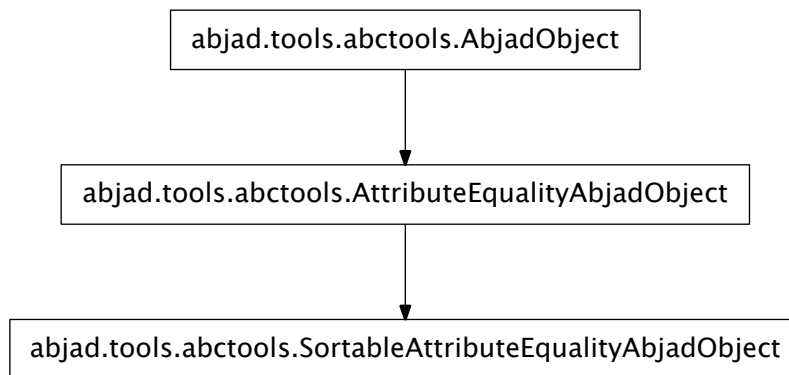
`ScoreSelection.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`abctools.SortableAttributeEqualityAbjadObject`



class `abctools.SortableAttributeEqualityAbjadObject(*args, **kwargs)`

New in version 2.0. Abstract base class to confer sortable attribute-equality to any custom class.

Sortable attribute-equality comparators implement `__eq__`, `__ne__`, `__gt__`, `__ge__`, `__lt__`, `__le__` against a single comparison attribute.

Read-only Properties

`SortableAttributeEqualityAbjadObject.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`SortableAttributeEqualityAbjadObject.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Inherited from `abctools.AttributeEqualityAbjadObject`
`SortableAttributeEqualityAbjadObject.__ge__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

`SortableAttributeEqualityAbjadObject.__gt__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

`SortableAttributeEqualityAbjadObject.__le__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

`SortableAttributeEqualityAbjadObject.__lt__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

`SortableAttributeEqualityAbjadObject.__ne__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

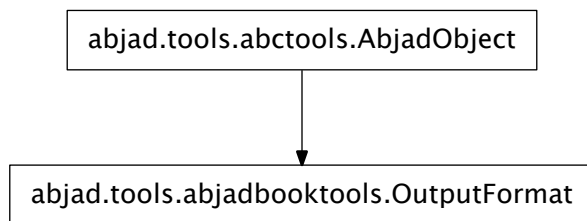
Inherited from `abctools.AttributeEqualityAbjadObject`
`SortableAttributeEqualityAbjadObject.__repr__()`
 Interpreter representation of object defined equal to class name and format string.
 Return string.

Inherited from `abctools.AttributeEqualityAbjadObject`

abjadbooktools

abstract classes

abjadbooktools.OutputFormat



class `abjadbooktools.OutputFormat` (*code_block_opening*, *code_block_closing*, *code_indent*, *image_block*, *image_format*)

Read-only Properties

`OutputFormat.code_block_closing`

`OutputFormat.code_block_opening`

`OutputFormat.code_indent`

`OutputFormat.image_block`

`OutputFormat.image_format`

`OutputFormat.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`OutputFormat.__call__(code_block)`

`OutputFormat.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`OutputFormat.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputFormat.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OutputFormat.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputFormat.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OutputFormat.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`OutputFormat.__repr__()`

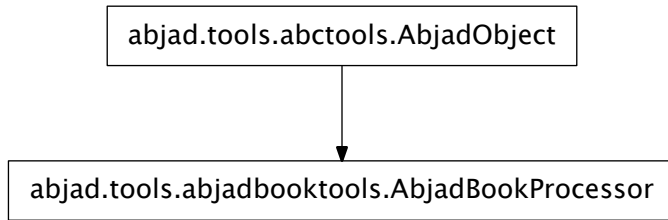
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

concrete classes

`abjadbooktools.AbjadBookProcessor`



```
class abjadbooktools.AbjadBookProcessor (directory, lines, output_format, skip_rendering=False,  
                                           image_prefix='image')
```

Read-only Properties

`AbjadBookProcessor.directory`

`AbjadBookProcessor.image_prefix`

`AbjadBookProcessor.lines`

`AbjadBookProcessor.output_format`

`AbjadBookProcessor.skip_rendering`

`AbjadBookProcessor.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`AbjadBookProcessor.__call__` (*verbose=True*)

`AbjadBookProcessor.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`AbjadBookProcessor.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadBookProcessor.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`AbjadBookProcessor.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadBookProcessor.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadBookProcessor.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

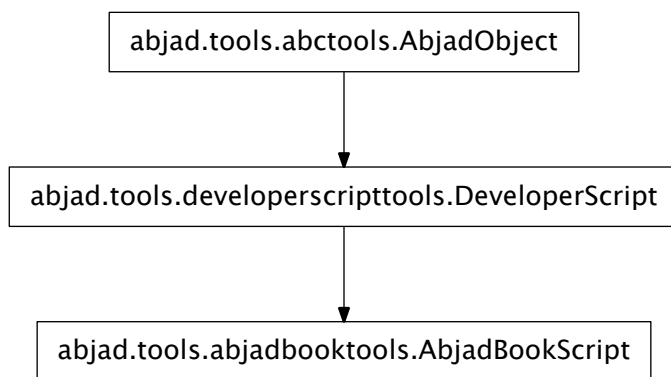
`AbjadBookProcessor.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`abjadbooktools.AbjadBookScript`



```
class abjadbooktools.AbjadBookScript
```

Read-only Properties

`AbjadBookScript.alias`

`AbjadBookScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscriptttools.DeveloperScript`

`AbjadBookScript.formatted_help`

Inherited from `developerscriptttools.DeveloperScript`

`AbjadBookScript.formatted_usage`

Inherited from `developerscriptttools.DeveloperScript`

`AbjadBookScript.formatted_version`

Inherited from `developerscriptttools.DeveloperScript`

`AbjadBookScript.long_description`

`AbjadBookScript.output_formats`

`AbjadBookScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

`AbjadBookScript.scripting_group`

The script's scripting subcommand group.

Inherited from `developerscriptttools.DeveloperScript`

`AbjadBookScript.short_description`

`AbjadBookScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`AbjadBookScript.version`

Methods

`AbjadBookScript.process_args(args)`

`AbjadBookScript.setup_argument_parser(parser)`

Special Methods

`AbjadBookScript.__call__(args=None)`

Inherited from `developerscriptttools.DeveloperScript`

`AbjadBookScript.__eq__(arg)`

True when `id(self) equals id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`AbjadBookScript.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadBookScript.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`AbjadBookScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadBookScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadBookScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

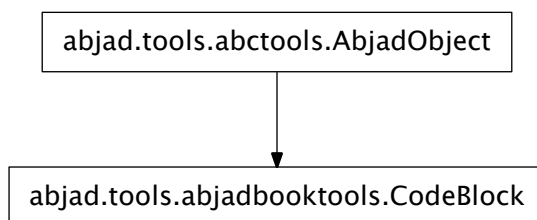
`AbjadBookScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

abjadbooktools.CodeBlock



```
class abjadbooktools.CodeBlock (lines, starting_line_number, ending_line_number, hide=False,
                                strip_prompt=False)
```

Read-only Properties

`CodeBlock.ending_line_number`

`CodeBlock.hide`

`CodeBlock.lines`

`CodeBlock.processed_results`

`CodeBlock.starting_line_number`

`CodeBlock.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`CodeBlock.strip_prompt`

Methods

`CodeBlock.read(pipe)`

Special Methods

`CodeBlock.__call__(pipe, image_count=0, directory=None, image_prefix='image')`

`CodeBlock.__eq__(other)`

`CodeBlock.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CodeBlock.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`CodeBlock.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CodeBlock.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CodeBlock.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

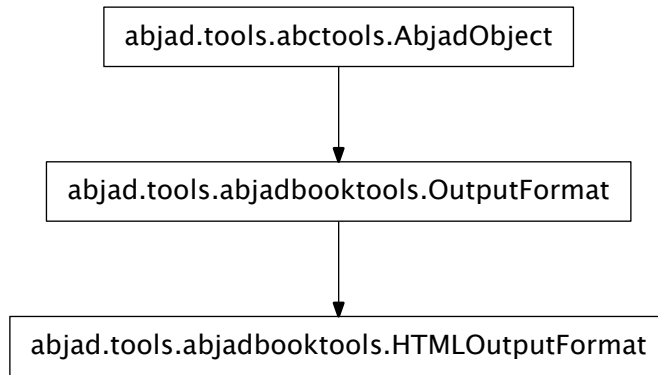
`CodeBlock.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

abjadbooktools.HTMLOutputFormat



class abjadbooktools.**HTMLOutputFormat**

Read-only Properties

HTMLOutputFormat.**code_block_closing**

Inherited from abjadbooktools.OutputFormat

HTMLOutputFormat.**code_block_opening**

Inherited from abjadbooktools.OutputFormat

HTMLOutputFormat.**code_indent**

Inherited from abjadbooktools.OutputFormat

HTMLOutputFormat.**image_block**

Inherited from abjadbooktools.OutputFormat

HTMLOutputFormat.**image_format**

Inherited from abjadbooktools.OutputFormat

HTMLOutputFormat.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from abctools.AbjadObject

Special Methods

HTMLOutputFormat.**__call__**(*code_block*)

Inherited from abjadbooktools.OutputFormat

HTMLOutputFormat.**__eq__**(*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from abctools.AbjadObject

HTMLOutputFormat.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HTMLOutputFormat.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`HTMLOutputFormat.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HTMLOutputFormat.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`HTMLOutputFormat.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

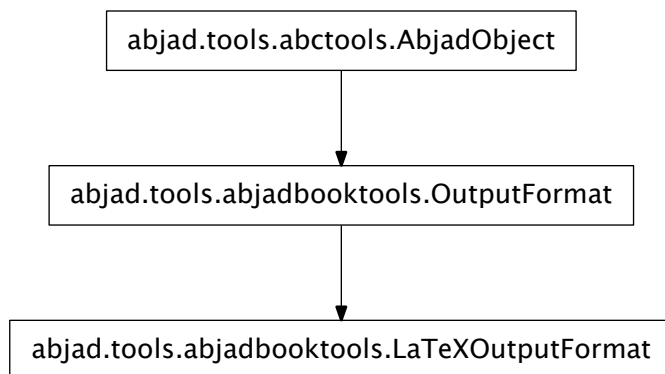
`HTMLOutputFormat.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`abjadbooktools.LaTeXOutputFormat`



```
class abjadbooktools.LaTeXOutputFormat
```

Read-only Properties

`LaTeXOutputFormat.code_block_closing`

Inherited from `abjadbooktools.OutputFormat`

`LaTeXOutputFormat.code_block_opening`

Inherited from `abjadbooktools.OutputFormat`

`LaTeXOutputFormat.code_indent`

Inherited from `abjadbooktools.OutputFormat`

`LaTeXOutputFormat.image_block`

Inherited from `abjadbooktools.OutputFormat`

`LaTeXOutputFormat.image_format`

Inherited from `abjadbooktools.OutputFormat`

`LaTeXOutputFormat.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`LaTeXOutputFormat.__call__` (*code_block*)

Inherited from `abjadbooktools.OutputFormat`

`LaTeXOutputFormat.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`LaTeXOutputFormat.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LaTeXOutputFormat.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`LaTeXOutputFormat.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LaTeXOutputFormat.__lt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LaTeXOutputFormat.__ne__` (*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

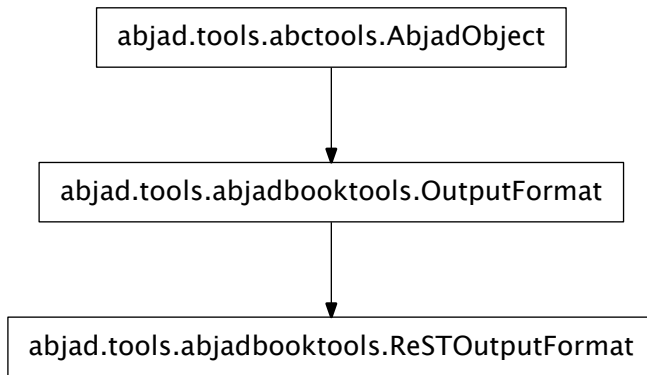
`LaTeXOutputFormat.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`abjadbooktools.ReSTOutputFormat`



class `abjadbooktools.ReSTOutputFormat`

Read-only Properties

`ReSTOutputFormat.code_block_closing`

Inherited from `abjadbooktools.OutputFormat`

`ReSTOutputFormat.code_block_opening`

Inherited from `abjadbooktools.OutputFormat`

`ReSTOutputFormat.code_indent`

Inherited from `abjadbooktools.OutputFormat`

`ReSTOutputFormat.image_block`

Inherited from `abjadbooktools.OutputFormat`

`ReSTOutputFormat.image_format`

Inherited from `abjadbooktools.OutputFormat`

`ReSTOutputFormat.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ReSTOutputFormat.__call__(code_block)`

Inherited from `abjadbooktools.OutputFormat`

`ReSTOutputFormat.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ReSTOutputFormat.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReSTOutputFormat.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ReSTOutputFormat.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReSTOutputFormat.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReSTOutputFormat.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ReSTOutputFormat.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

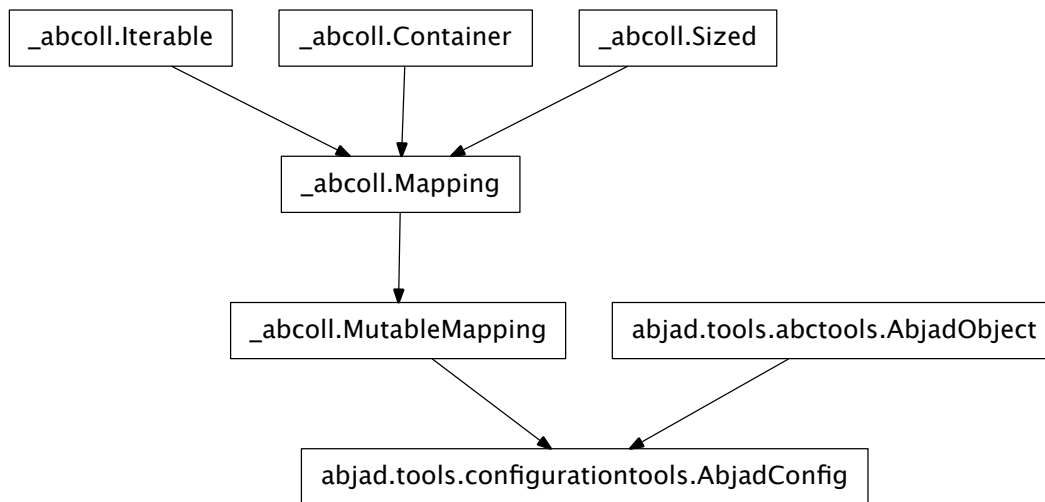
Return string.

Inherited from `abctools.AbjadObject`

configurationtools

concrete classes

configurationtools.AbjadConfig



class configurationtools.AbjadConfig

Abjad configuration object:

```

>>> from abjad.tools.configurationtools import AbjadConfig
>>> ABJCONFIG = AbjadConfig()
>>> ABJCONFIG['accidental_spelling']
'mixed'

```

On instantiation, *AbjadConfig* creates the *\$HOME/abjad/* directory if it does not already exist.

It then attempts to read an *abjad.cfg* file in that directory, parsing it as a *ConfigObj* configuration. A default configuration is generated if no file is found.

The *ConfigObj* instance is validated, and key:value pairs which fail validation are replaced by default values.

The configuration is then written back to disk.

Finally, the Abjad output directory is created if it does not already exist, by referencing the 'abjad_output' key in the configuration.

AbjadConfig supports the mutable mapping interface, and can be subscripted as a dictionary.

Returns *AbjadConfig* instance.

Read-only Properties

AbjadConfig.**ABJAD_CONFIG_DIRECTORY_PATH**

AbjadConfig.**ABJAD_CONFIG_FILE_PATH**

AbjadConfig.**ABJAD_EXPERIMENTAL_PATH**

AbjadConfig.**ABJAD_OUTPUT_PATH**

AbjadConfig.**ABJAD_PATH**

AbjadConfig.**ABJAD_ROOT_PATH**

AbjadConfig.**HOME_PATH**

AbjadConfig.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

AbjadConfig.**clear**()

Inherited from `_abcoll.MutableMapping`

AbjadConfig.**get**(key, default=None)

Inherited from `_abcoll.Mapping`

AbjadConfig.**get_config_spec**()

AbjadConfig.**get_initial_comment**()

AbjadConfig.**get_option_comments**()

AbjadConfig.**get_option_definitions**()

AbjadConfig.**get_option_specs**()

AbjadConfig.**items**()

Inherited from `_abcoll.Mapping`

AbjadConfig.**iteritems**()

Inherited from `_abcoll.Mapping`

AbjadConfig.**iterkeys**()

Inherited from `_abcoll.Mapping`

AbjadConfig.**itervalues**()

Inherited from `_abcoll.Mapping`

AbjadConfig.**keys**()

Inherited from `_abcoll.Mapping`

AbjadConfig.**pop**(key, default=<object object at 0x1002b0040>)

Inherited from `_abcoll.MutableMapping`

AbjadConfig.**popitem**()

Inherited from `_abcoll.MutableMapping`

AbjadConfig.**setdefault**(key, default=None)

Inherited from `_abcoll.MutableMapping`

AbjadConfig.**update**(*args, **kws)

Inherited from `_abcoll.MutableMapping`

AbjadConfig.**values**()

Inherited from `_abcoll.Mapping`

Special Methods

`AbjadConfig.__contains__(key)`

Inherited from `_abcoll.Mapping`

`AbjadConfig.__delitem__(i)`

`AbjadConfig.__eq__(other)`

Inherited from `_abcoll.Mapping`

`AbjadConfig.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadConfig.__getitem__(i)`

`AbjadConfig.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`AbjadConfig.__iter__()`

`AbjadConfig.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadConfig.__len__()`

`AbjadConfig.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjadConfig.__ne__(other)`

Inherited from `_abcoll.Mapping`

`AbjadConfig.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`AbjadConfig.__setitem__(i, arg)`

functions

`configurationtools.get_abjad_revision_string`

`configurationtools.get_abjad_revision_string()`

New in version 2.0. Get Abjad revision string:

```
>>> configurationtools.get_abjad_revision_string()
'5280'
```

Return string.

`configurationtools.get_abjad_startup_string`

```
configurationtools.get_abjad_startup_string()
```

`configurationtools.get_abjad_version_string`

```
configurationtools.get_abjad_version_string()
```

New in version 2.0. Get Abjad version string:

```
>>> from abjad.tools import configurationtools

>>> configurationtools.get_abjad_version_string()
'2.10'
```

Return string.

`configurationtools.get_lilypond_version_string`

```
configurationtools.get_lilypond_version_string()
```

New in version 2.0. Get LilyPond version string:

```
>>> configurationtools.get_lilypond_version_string()
'2.13.61'
```

Return string.

`configurationtools.get_python_version_string`

```
configurationtools.get_python_version_string()
```

New in version 2.0. Get Python version string:

```
>>> from abjad.tools import configurationtools

>>> configurationtools.get_python_version_string()
'2.6.1'
```

Return string.

`configurationtools.get_tab_width`

```
configurationtools.get_tab_width()
```

New in version 2.9. Get system tab width:

```
>>> from abjad.tools import configurationtools

>>> configurationtools.get_tab_width()
4
```

The value is used by various functions that generate or test code in the system.

Return nonnegative integer. Changed in version 2.10: renamed `configurationtools.get_system_tab_width()` to `configurationtools.get_tab_width()`.

configurationtools.get_text_editor

`configurationtools.get_text_editor()`
New in version 2.2. Get OS-appropriate text editor.

configurationtools.list_abjad_environment_variables

`configurationtools.list_abjad_environment_variables()`
New in version 1.1. List Abjad environment variables.

Return tuple of zero or more environment variable / setting pairs.

Abjad environment variables are defined in `abjad/tools/configurationtools/AbjadConfig/AbjadConfig.py`.
Changed in version 2.0: renamed `configurationtools.list_settings()` to `configurationtools.list_abjad_environment_variables()`.

configurationtools.list_package_dependency_versions

`configurationtools.list_package_dependency_versions()`
List package dependency versions:

```
>>> from abjad.tools import configurationtools

>>> configurationtools.list_package_dependency_versions()
{'sphinx': '1.1.2', 'py.test': '2.1.2'}
```

Return dictionary.

configurationtools.read_abjad_user_config_file

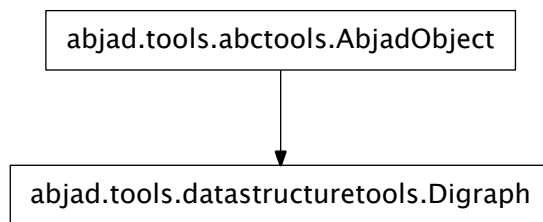
`configurationtools.read_abjad_user_config_file(attribute_name)`
Read the content of the config file `$HOME/.abjad/config.py`.

Returns a dictionary of var : value entries. Changed in version 2.10: renamed `configurationtools.read_user_abjad_config_file()` to `ABJCFG()`.

datastructuretools

concrete classes

datastructuretools.Digraph



class datastructuretools.**Digraph**(edges=None)

A digraph, built out of edges - pairs of hashable objects:

```

>>> from abjad.tools.datastructuretools import Digraph

>>> edges = [('a', 'b'), ('a', 'c'), ('a', 'f'), ('c', 'd'), ('d', 'e'), ('e', 'c')]
>>> digraph = Digraph(edges)
>>> digraph
Digraph(edges=[('a', 'c'), ('a', 'b'), ('a', 'f'), ('c', 'd'), ('d', 'e'), ('e', 'c')])

>>> digraph.root_nodes
('a',)
>>> digraph.terminal_nodes
('b', 'f')
>>> digraph.cyclic_nodes
('c', 'd', 'e')
>>> digraph.is_cyclic
True
  
```

Return *Digraph* instance.

Read-only Properties

Digraph.child_graph

A dictionary representation of the digraph where the keys are child nodes, and where each value is the set of that child's parents.

Digraph.cyclic_nodes

A tuple of those nodes which partake in a cycle.

Digraph.edges

A tuple of all edges in the graph.

Digraph.is_cyclic

Return True if the digraph contains any cycles.

Digraph.nodes

A tuple of all nodes in the graph.

Digraph.parent_graph

A dictionary representation of the digraph where the keys are parent nodes, and where each value is the set of that parent's children.

Digraph.root_nodes

A tuple of those nodes which have no parents.

Digraph.storage_format

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Digraph.terminal_nodes

A tuple of those nodes which have no children.

Methods**Digraph.partition()**

Partition the digraph into a list of digraphs according to connectivity:

```
>>> from abjad.tools.datastructuretools import Digraph

>>> edges = [('a', 'b'), ('a', 'c'), ('b', 'c'), ('b', 'd'), ('d', 'e')]
>>> edges.extend([('f', 'h'), ('g', 'h')])
>>> edges.append(('i', 'j'))
>>> digraph = Digraph(edges)
>>> for graph in digraph.partition(): graph
...
Digraph(edges=[('a', 'c'), ('a', 'b'), ('b', 'c'), ('b', 'd'), ('d', 'e')])
Digraph(edges=[('f', 'h'), ('g', 'h')])
Digraph(edges=[('i', 'j')])
```

Return list of *Digraph* instances.

Digraph.reverse()

Reverse all edges in the graph:

```
>>> from abjad.tools.datastructuretools import Digraph

>>> edges = [('a', 'b'), ('b', 'c'), ('b', 'd')]
>>> digraph = Digraph(edges)
>>> digraph
Digraph(edges=[('a', 'b'), ('b', 'c'), ('b', 'd')])

>>> digraph.reverse()
Digraph(edges=[('b', 'a'), ('c', 'b'), ('d', 'b')])
```

Return *Digraph* instance.

Special Methods**Digraph.__eq__(other)****Digraph.__ge__(arg)**

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Digraph.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

Digraph.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Digraph.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

Digraph.__ne__(other)

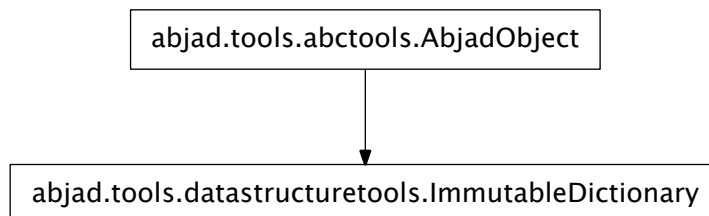
Digraph.__repr__()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`datastructuretools.ImmutableDictionary`



class `datastructuretools.ImmutableDictionary`

New in version 2.0. Immutable dictionary:

```
>>> from abjad.tools import datastructuretools
```

```
>>> dictionary = datastructuretools.ImmutableDictionary({'color': 'red', 'number': 9})
```

```
>>> dictionary
{'color': 'red', 'number': 9}
```

```
>>> dictionary['color']
'red'
```

```
>>> dictionary.size = 'large'
AttributeError: ImmutableDictionary objects are immutable.
```

```
>>> dictionary['size'] = 'large'
AttributeError: ImmutableDictionary objects are immutable.
```

Return immutable dictionary.

Read-only Properties

`ImmutableDictionary.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`ImmutableDictionary.clear()` → None. Remove all items from D.

Inherited from `__builtin__.dict`

`ImmutableDictionary.copy()` → a shallow copy of D

Inherited from `__builtin__.dict`

`ImmutableDictionary.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Inherited from `__builtin__.dict`

`ImmutableDictionary.has_key(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`ImmutableDictionary.items()` → list of D's (key, value) pairs, as 2-tuples

Inherited from `__builtin__.dict`

`ImmutableDictionary.iteritems()` → an iterator over the (key, value) items of D

Inherited from `__builtin__.dict`

`ImmutableDictionary.iterkeys()` → an iterator over the keys of D

Inherited from `__builtin__.dict`

`ImmutableDictionary.itervalues()` → an iterator over the values of D

Inherited from `__builtin__.dict`

`ImmutableDictionary.keys()` → list of D's keys

Inherited from `__builtin__.dict`

`ImmutableDictionary.pop(k[, d])` → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

Inherited from `__builtin__.dict`

`ImmutableDictionary.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Inherited from `__builtin__.dict`

`ImmutableDictionary.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Inherited from `__builtin__.dict`

`ImmutableDictionary.update([E], **F)` → None. Update D from dict/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Inherited from `__builtin__.dict`

`ImmutableDictionary.values()` → list of D's values

Inherited from `__builtin__.dict`

`ImmutableDictionary.viewitems()` → a set-like object providing a view on D's items

Inherited from `__builtin__.dict`

`ImmutableDictionary.viewkeys()` → a set-like object providing a view on D's keys

Inherited from `__builtin__.dict`

`ImmutableDictionary.viewvalues()` → an object providing a view on D's values

Inherited from `__builtin__.dict`

Special Methods

`ImmutableDictionary.__cmp__(y) <==> cmp(x, y)`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__contains__(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`ImmutableDictionary.__delitem__(*args)`

`ImmutableDictionary.__eq__()`

`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__ge__()`

`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__gt__()`

`x.__gt__(y) <==> x>y`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__iter__()` <==> `iter(x)`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__le__()`

`x.__le__(y) <==> x<=y`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__len__()` <==> `len(x)`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__lt__()`

`x.__lt__(y) <==> x<y`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__ne__()`

`x.__ne__(y) <==> x!=y`

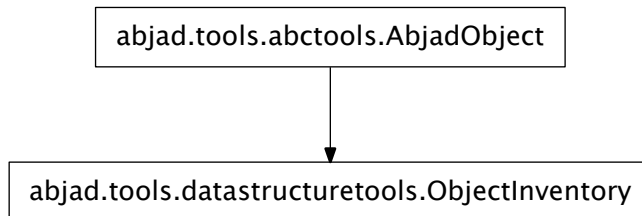
Inherited from `__builtin__.dict`

`ImmutableDictionary.__repr__()` <==> `repr(x)`

Inherited from `__builtin__.dict`

`ImmutableDictionary.__setitem__(*args)`

`datastructuretools.ObjectInventory`



class `datastructuretools.ObjectInventory` (*tokens=None, name=None*)

New in version 2.8. Ordered collection of custom objects.

Object inventories extend `append()`, `extend()` and `__contains__()` and allow token input.

Object inventories inherit from list and are mutable.

This class is an abstract base class that can not instantiate and should be subclassed.

Read-only Properties

`ObjectInventory.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`ObjectInventory.name`

Read / write name of inventory.

Methods

`ObjectInventory.append(token)`

Change *token* to item and append.

`ObjectInventory.count(value)` → integer – return number of occurrences of value

Inherited from `__builtin__.list`

`ObjectInventory.extend(tokens)`

Change *tokens* to items and extend.

`ObjectInventory.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`ObjectInventory.insert()`

`L.insert(index, object)` – insert object before index

Inherited from `__builtin__.list`

`ObjectInventory.pop([index])` → item – remove and return item at index (default last).
Raises `IndexError` if list is empty or index is out of range.

Inherited from `__builtin__.list`

`ObjectInventory.remove()`
`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Inherited from `__builtin__.list`

`ObjectInventory.reverse()`
`L.reverse()` – reverse *IN PLACE*

Inherited from `__builtin__.list`

`ObjectInventory.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Inherited from `__builtin__.list`

Special Methods

`ObjectInventory.__add__()`
`x.__add__(y) <==> x+y`

Inherited from `__builtin__.list`

`ObjectInventory.__contains__(token)`

`ObjectInventory.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Inherited from `__builtin__.list`

`ObjectInventory.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
Use of negative indices is not supported.

Inherited from `__builtin__.list`

`ObjectInventory.__eq__()`
`x.__eq__(y) <==> x==y`

Inherited from `__builtin__.list`

`ObjectInventory.__ge__()`
`x.__ge__(y) <==> x>=y`

Inherited from `__builtin__.list`

`ObjectInventory.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Inherited from `__builtin__.list`

`ObjectInventory.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
Use of negative indices is not supported.

Inherited from `__builtin__.list`

```

ObjectInventory.__gt__()
    x.__gt__(y) <==> x>y

    Inherited from __builtin__.list

ObjectInventory.__iadd__()
    x.__iadd__(y) <==> x+=y

    Inherited from __builtin__.list

ObjectInventory.__imul__()
    x.__imul__(y) <==> x*=y

    Inherited from __builtin__.list

ObjectInventory.__iter__() <==> iter(x)
    Inherited from __builtin__.list

ObjectInventory.__le__()
    x.__le__(y) <==> x<=y

    Inherited from __builtin__.list

ObjectInventory.__len__() <==> len(x)
    Inherited from __builtin__.list

ObjectInventory.__lt__()
    x.__lt__(y) <==> x<y

    Inherited from __builtin__.list

ObjectInventory.__mul__()
    x.__mul__(n) <==> x*n

    Inherited from __builtin__.list

ObjectInventory.__ne__()
    x.__ne__(y) <==> x!=y

    Inherited from __builtin__.list

ObjectInventory.__repr__()

ObjectInventory.__reversed__()
    L.__reversed__() – return a reverse iterator over the list

    Inherited from __builtin__.list

ObjectInventory.__rmul__()
    x.__rmul__(n) <==> n*x

    Inherited from __builtin__.list

ObjectInventory.__setitem__()
    x.__setitem__(i, y) <==> x[i]=y

    Inherited from __builtin__.list

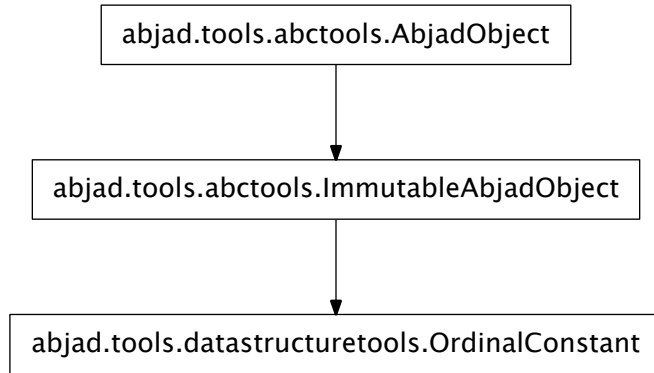
ObjectInventory.__setslice__()
    x.__setslice__(i, j, y) <==> x[i:j]=y

    Use of negative indices is not supported.

    Inherited from __builtin__.list

```

datastructuretools.OrdinalConstant



class datastructuretools.**OrdinalConstant** (*args, **kwargs)
 New in version 1.0. Ordinal constant.

Initialize with *dimension*, *value* and *representation*:

```
>>> from abjad.tools import datastructuretools

>>> Left = datastructuretools.OrdinalConstant('x', -1, 'Left')
>>> Left
Left

>>> Right = datastructuretools.OrdinalConstant('x', 1, 'Right')
>>> Right
Right

>>> Left < Right
True
```

Comparing like-dimensioned ordinal constants is allowed:

```
>>> Up = datastructuretools.OrdinalConstant('y', 1, 'Up')
>>> Up
Up

>>> Down = datastructuretools.OrdinalConstant('y', -1, 'Down')
>>> Down
Down

>>> Down < Up
True
```

Comparing differently dimensioned ordinal constants raises an exception:

```
>>> import py.test
```

```
>>> bool(py.test.raises(Exception, 'Left < Up'))
True
```

The Left, Right, Center, Up and Down constants shown here load into Python’s built-in namespace on Abjad import.

These four objects can be used as constant values supplied to keywords.

This behavior is similar to True, False and None.

Ordinal constants are immutable.

Read-only Properties

`OrdinalConstant.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`OrdinalConstant.__eq__(expr)`

`OrdinalConstant.__ge__(expr)`

`OrdinalConstant.__gt__(expr)`

`OrdinalConstant.__le__(expr)`

`OrdinalConstant.__lt__(expr)`

`OrdinalConstant.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`OrdinalConstant.__repr__()`

decoratortools

functions

decoratortools.requires

`decoratortools.requires(*tests)`

New in version 2.6. Function decorator to require input parameter *tests*.

Example:

```
>>> from abjad.tools import mathtools
>>> from abjad.tools.decoratortools import requires

>>> @requires(mathtools.is_nonnegative_integer, string)
>>> def multiply_string(n, string): return n * string

>>> multiply_string(2, 'bar')
'barbar'
```

```
>>> multiply_string(2.5, 'bar')
...
AssertionError: is_nonnegative_integer(2.5) does not return true.
```

Decorator target is available like this:

```
>>> multiply_string.func_closure[1].cell_contents
<function multiply_string at 0x104e512a8>
```

Decorator tests are available like this:

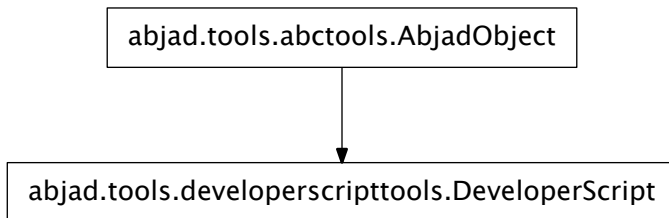
```
>>> multiply_string.func_closure[0].cell_contents
(<function is_nonnegative_integer at 0x104725d70>, <type 'str'>)
```

Return decorated function in the form of function wrapper.

developerscripttools

abstract classes

developerscripttools.DeveloperScript



class developerscripttools.DeveloperScript

Abjad object-oriented model of a developer script.

DeveloperScript is the abstract parent from which concrete developer scripts inherit.

Developer scripts can be called from the command line, generally via the *ajv* command.

Developer scripts can be instantiated by other developer scripts in order to share functionality.

Read-only Properties

`DeveloperScript.alias`

The alias to use for the script, useful only if the script defines an abj-dev scripting group as well.

`DeveloperScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

`DeveloperScript.formatted_help`

`DeveloperScript.formatted_usage`

`DeveloperScript.formatted_version`

`DeveloperScript.long_description`

The long description, printed after arguments explanations.

`DeveloperScript.program_name`

The name of the script, callable from the command line.

`DeveloperScript.scripting_group`

The script's scripting subcommand group.

`DeveloperScript.short_description`

The short description of the script, printed before arguments explanations.

Also used as a summary in other contexts.

`DeveloperScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`DeveloperScript.version`

The version number of the script.

Methods

`DeveloperScript.process_args(args)`

`DeveloperScript.setup_argument_parser()`

Special Methods

`DeveloperScript.__call__(args=None)`

`DeveloperScript.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DeveloperScript.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DeveloperScript.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DeveloperScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DeveloperScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DeveloperScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

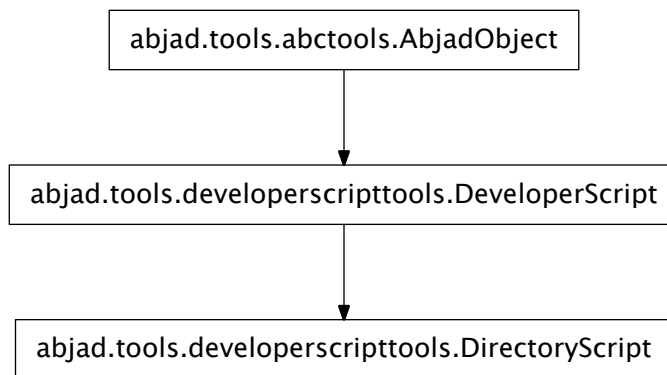
`DeveloperScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`developerscripttools.DirectoryScript`



class `developerscripttools.DirectoryScript`

DirectoryScript provides utilities for validating file system paths.

DirectoryScript is abstract.

Read-only Properties

`DirectoryScript.alias`

The alias to use for the script, useful only if the script defines an abj-dev scripting group as well.

Inherited from `developerscripttools.DeveloperScript`

`DirectoryScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscripttools.DeveloperScript`

`DirectoryScript.formatted_help`

Inherited from `developerscripttools.DeveloperScript`

`DirectoryScript.formatted_usage`

Inherited from `developerscripttools.DeveloperScript`

`DirectoryScript.formatted_version`

Inherited from `developerscripttools.DeveloperScript`

`DirectoryScript.long_description`

The long description, printed after arguments explanations.

Inherited from `developerscriptttools.DeveloperScript`

`DirectoryScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

`DirectoryScript.scripting_group`

The script's scripting subcommand group.

Inherited from `developerscriptttools.DeveloperScript`

`DirectoryScript.short_description`

The short description of the script, printed before arguments explanations.

Also used as a summary in other contexts.

Inherited from `developerscriptttools.DeveloperScript`

`DirectoryScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`DirectoryScript.version`

The version number of the script.

Inherited from `developerscriptttools.DeveloperScript`

Methods

`DirectoryScript.process_args(args)`

Inherited from `developerscriptttools.DeveloperScript`

`DirectoryScript.setup_argument_parser()`

Inherited from `developerscriptttools.DeveloperScript`

Special Methods

`DirectoryScript.__call__(args=None)`

Inherited from `developerscriptttools.DeveloperScript`

`DirectoryScript.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DirectoryScript.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DirectoryScript.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DirectoryScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DirectoryScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DirectoryScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DirectoryScript.__repr__()`

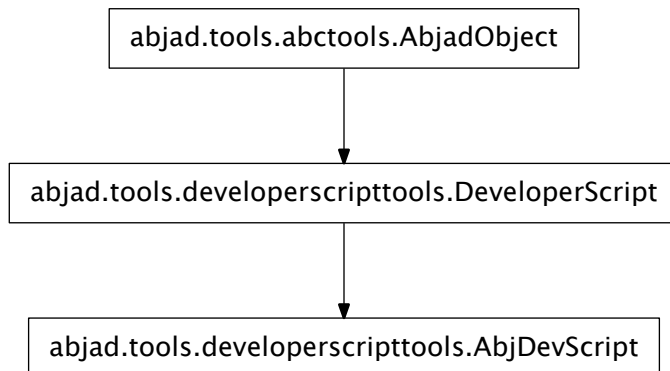
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

concrete classes

`developerscripttools.AbjDevScript`



class `developerscripttools.AbjDevScript`

AbjDevScript is the commandline entry-point to the Abjad developer scripts catalog.

Can be accessed on the commandline via *abj-dev* or *ajv*:

```
bash$ abj-dev
usage: abj-dev [-h] [--version]
```

```
{help, list, api, book, clean, count, doctest, grep, new, rename, replace, svn}
...
```

Entry-point to Abjad developer scripts catalog.

optional arguments:

```
-h, --help            show this help message and exit
--version             show program's version number and exit
```

subcommands:

```
{help, list, api, book, clean, count, doctest, grep, new, rename, replace, svn}
  help                print subcommand help
  list                list subcommands
  api                 Build the Abjad APIs.
  book                Preprocess HTML, LaTeX or ReST source with Abjad.
  clean               Clean .pyc, __pycache__ and tmp* files and folders
                     from PATH.
  count               "count"-related subcommands
  doctest             Run doctests on all modules in current path.
  grep                grep PATTERN in PATH
  new                 "new"-related subcommands
  rename              Rename public modules.
  replace             "replace"-related subcommands
  svn                 "svn"-related subcommands
```

ajv supports subcommands similar to *svn*:

```
bash$ ajv count -h
usage: abj-dev count [-h] {linewidths,tools} ...
```

optional arguments:

```
-h, --help            show this help message and exit
```

count subcommands:

```
{linewidths,tools}
  linewidths          Count maximum line-width of all modules in PATH.
  tools                Count tools in PATH.
```

Return *AbjDevScript* instance.

Read-only Properties

AbjDevScript.**alias**

The alias to use for the script, useful only if the script defines an abj-dev scripting group as well.

Inherited from *developerscripttools*.*DeveloperScript*

AbjDevScript.**argument_parser**

The script's instance of *argparse*.*ArgumentParser*.

Inherited from *developerscripttools*.*DeveloperScript*

AbjDevScript.**developer_script_aliases**

AbjDevScript.**developer_script_classes**

AbjDevScript.**developer_script_program_names**

AbjDevScript.**formatted_help**

Inherited from *developerscripttools*.*DeveloperScript*

`AbjDevScript.formatted_usage`

Inherited from `developerscripttools.DeveloperScript`

`AbjDevScript.formatted_version`

Inherited from `developerscripttools.DeveloperScript`

`AbjDevScript.long_description`

`AbjDevScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscripttools.DeveloperScript`

`AbjDevScript.scripting_group`

The script's scripting subcommand group.

Inherited from `developerscripttools.DeveloperScript`

`AbjDevScript.short_description`

`AbjDevScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`AbjDevScript.version`

Methods

`AbjDevScript.process_args` (*args*)

`AbjDevScript.setup_argument_parser` (*parser*)

Special Methods

`AbjDevScript.__call__` (*args=None*)

`AbjDevScript.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`AbjDevScript.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjDevScript.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`AbjDevScript.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjDevScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`AbjDevScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

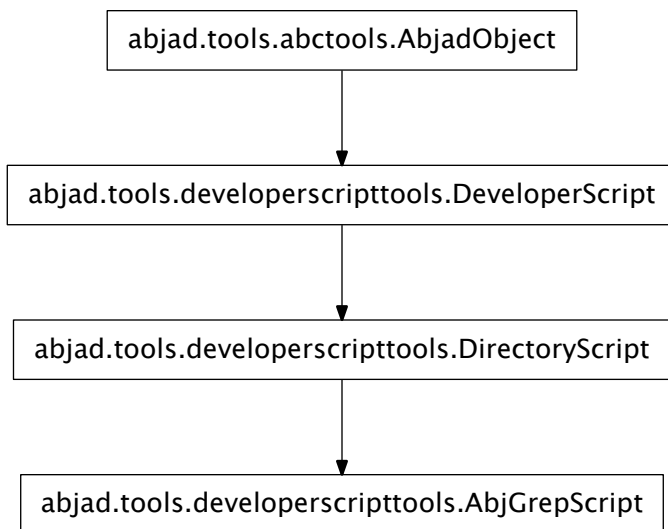
`AbjDevScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`developerscripttools.AbjGrepScript`



class `developerscripttools.AbjGrepScript`

Run *grep* against a path, ignoring *svn* and docs-related files:

```
bash$ ajv grep -h
usage: abj-grep [-h] [--version] [-W] [-P PATH | -X | -M | -T | -R] pattern
```

```
grep PATTERN in PATH
```

positional arguments:

pattern pattern to search for

optional arguments:

-h, --help	show this help message and exit
--version	show program's version number and exit
-W, --whole-words-only	match only whole words, similar to grep's "-w" flag
-P PATH, --path PATH	grep PATH
-X, --abjad.tools	grep Abjad abjad.tools directory
-M, --mainline	grep Abjad mainline directory
-T, --tools	grep Abjad mainline tools directory
-R, --root	grep Abjad root directory

If no PATH flag is specified, the current directory will be searched.

Return *AbjGrepScript* instance.

Read-only Properties

`AbjGrepScript.alias`

`AbjGrepScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscriptttools.DeveloperScript`

`AbjGrepScript.formatted_help`

Inherited from `developerscriptttools.DeveloperScript`

`AbjGrepScript.formatted_usage`

Inherited from `developerscriptttools.DeveloperScript`

`AbjGrepScript.formatted_version`

Inherited from `developerscriptttools.DeveloperScript`

`AbjGrepScript.long_description`

`AbjGrepScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

`AbjGrepScript.scripting_group`

`AbjGrepScript.short_description`

`AbjGrepScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`AbjGrepScript.version`

Methods

`AbjGrepScript.process_args(args)`

`AbjGrepScript.setup_argument_parser(parser)`

Special Methods

`AbjGrepScript.__call__(args=None)`

Inherited from `developerscriptttools.DeveloperScript`

AbjGrepScript.**__eq__**(arg)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

AbjGrepScript.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

AbjGrepScript.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

AbjGrepScript.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

AbjGrepScript.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

AbjGrepScript.**__ne__**(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

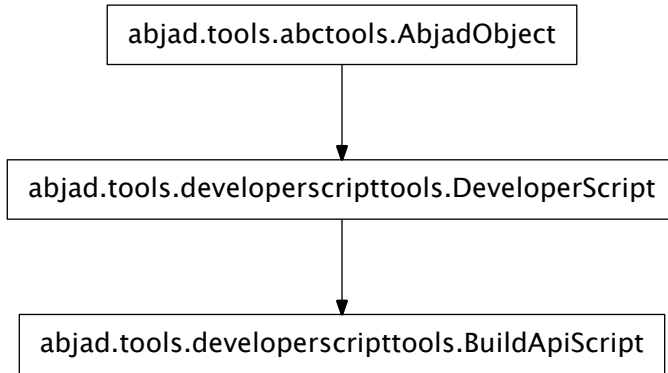
AbjGrepScript.**__repr__**()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

developerscripttools.BuildApiScript



class developerscripttools.**BuildApiScript**

Build the Abjad APIs:

```
bash$ ajv api -h
usage: build-api [-h] [--version] [-M] [-X] [-C] [--format FORMAT]
```

Build the Abjad APIs.

optional arguments:

-h, --help	show this help message and exit
--version	show program's version number and exit
-M, --mainline	build the mainline API
-X, --abjad.tools	build the abjad.tools API
-C, --clean	run "make clean" before building the api
--format FORMAT	Sphinx builder to use

Return *BuildApiScript* instance.

Read-only Properties

BuildApiScript.alias

BuildApiScript.argument_parser

The script's instance of argparse.ArgumentParser.

Inherited from developerscripttools.DeveloperScript

BuildApiScript.formatted_help

Inherited from developerscripttools.DeveloperScript

BuildApiScript.formatted_usage

Inherited from developerscripttools.DeveloperScript

BuildApiScript.formatted_version

Inherited from developerscripttools.DeveloperScript

BuildApiScript.long_description

`BuildApiScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscripttools.DeveloperScript`

`BuildApiScript.scripting_group`

`BuildApiScript.short_description`

`BuildApiScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`BuildApiScript.version`

Methods

`BuildApiScript.process_args` (*args*)

`BuildApiScript.setup_argument_parser` (*parser*)

Special Methods

`BuildApiScript.__call__` (*args=None*)

Inherited from `developerscripttools.DeveloperScript`

`BuildApiScript.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`BuildApiScript.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BuildApiScript.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BuildApiScript.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BuildApiScript.__lt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BuildApiScript.__ne__` (*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

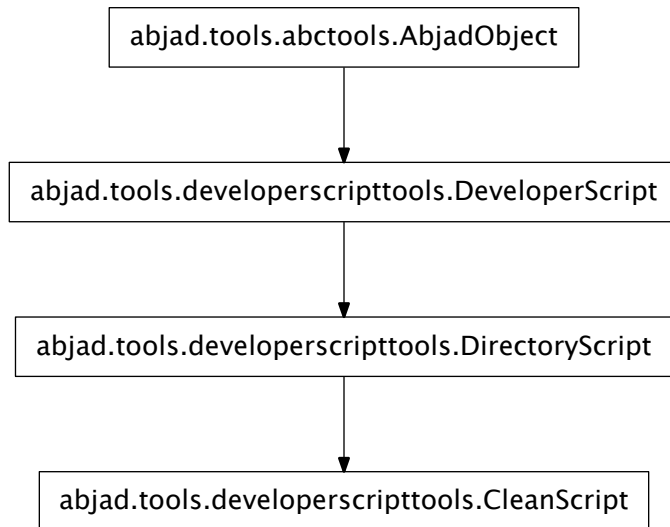
`BuildApiScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`developerscripttools.CleanScript`



class `developerscripttools.CleanScript`

Remove *.pyc* files and `__pycache__` and *tmp* directories recursively in a path:

```
bash$ ajv clean -h
```

```
usage: clean [-h] [--version] [--pyc] [--pycache] [--tmp] path
```

Clean *.pyc*, `__pycache__` and *tmp** files and folders from PATH.

positional arguments:

path directory tree to be recursed over

optional arguments:

```
-h, --help  show this help message and exit
--version  show program's version number and exit
--pyc      delete .pyc files
--pycache  delete __pycache__ folders
--tmp      delete tmp* folders
```

Return *CleanScript* instance.

Read-only Properties

`CleanScript.alias`

`CleanScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscriptttools.DeveloperScript`

`CleanScript.formatted_help`

Inherited from `developerscriptttools.DeveloperScript`

`CleanScript.formatted_usage`

Inherited from `developerscriptttools.DeveloperScript`

`CleanScript.formatted_version`

Inherited from `developerscriptttools.DeveloperScript`

`CleanScript.long_description`

`CleanScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

`CleanScript.scripting_group`

`CleanScript.short_description`

`CleanScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`CleanScript.version`

Methods

`CleanScript.process_args` (*args*)

`CleanScript.setup_argument_parser` (*parser*)

Special Methods

`CleanScript.__call__` (*args=None*)

Inherited from `developerscriptttools.DeveloperScript`

`CleanScript.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`CleanScript.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CleanScript.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`CleanScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CleanScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CleanScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

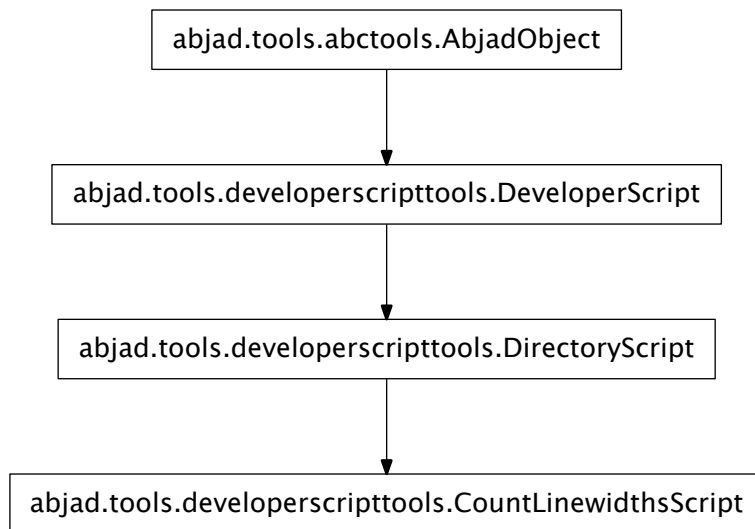
`CleanScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

developerscripttools.CountLinewidthsScript



class `developerscripttools.CountLinewidthsScript`

Tabulate the linewidths of modules in a path:

```
bash$ ajv count linewidths -h
usage: count-linewidths [-h] [--version] [-l N] [-o w|m] [-C | -D] [-a | -d]
                        [-gt N | -lt N | -eq N]
                        path
```

Count maximum line-width of all modules in PATH.

positional arguments:

path directory tree to be recursed over

optional arguments:

```
-h, --help            show this help message and exit
--version            show program's version number and exit
-l N, --limit N      limit output to last N items
-o w|m, --order-by w|m
                    order by line width [w] or module name [m]
-C, --code           count linewidths of all code in module
-D, --docstrings     count linewidths of all docstrings in module
-a, --ascending      sort results ascending
-d, --descending     sort results descending
-gt N, --greater-than N
                    line widths greater than N
-lt N, --less-than N line widths less than N
-eq N, --equal-to N  line widths equal to N
```

Return *CountLinewidthsScript* instance.

Read-only Properties

CountLinewidthsScript.**alias**

CountLinewidthsScript.**argument_parser**

The script's instance of *argparse.ArgumentParser*.

Inherited from *developerscripttools.DeveloperScript*

CountLinewidthsScript.**formatted_help**

Inherited from *developerscripttools.DeveloperScript*

CountLinewidthsScript.**formatted_usage**

Inherited from *developerscripttools.DeveloperScript*

CountLinewidthsScript.**formatted_version**

Inherited from *developerscripttools.DeveloperScript*

CountLinewidthsScript.**long_description**

CountLinewidthsScript.**program_name**

The name of the script, callable from the command line.

Inherited from *developerscripttools.DeveloperScript*

CountLinewidthsScript.**scripting_group**

CountLinewidthsScript.**short_description**

CountLinewidthsScript.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from *abctools.AbjadObject*

CountLinewidthsScript.**version**

Methods

CountLinewidthsScript.**process_args**(args)

CountLinewidthsScript.**setup_argument_parser**(parser)

Special Methods

CountLinewidthsScript.**__call__**(args=None)

Inherited from developerscripttools.DeveloperScript

CountLinewidthsScript.**__eq__**(arg)

True when id(self) equals id(arg).

Return boolean.

Inherited from abctools.AbjadObject

CountLinewidthsScript.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

CountLinewidthsScript.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from abctools.AbjadObject

CountLinewidthsScript.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

CountLinewidthsScript.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from abctools.AbjadObject

CountLinewidthsScript.**__ne__**(arg)

True when id(self) does not equal id(arg).

Return boolean.

Inherited from abctools.AbjadObject

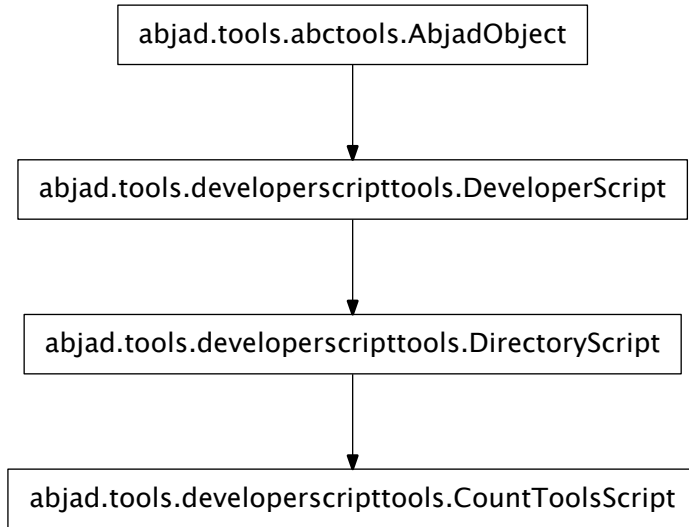
CountLinewidthsScript.**__repr__**()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from abctools.AbjadObject

developerscripttools.CountToolsScript



class `developerscripttools.CountToolsScript`

Count public and private functions and classes in a path:

```
bash$ ajv count tools -h
usage: count-tools [-h] [--version] path
```

Count tools in PATH.

positional arguments:

path directory tree to be recursed over

optional arguments:

-h, --help show this help message and exit
 --version show program's version number and exit

Return *CountToolsScript* instance.

Read-only Properties

`CountToolsScript.alias`

`CountToolsScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscripttools.DeveloperScript`

`CountToolsScript.formatted_help`

Inherited from `developerscripttools.DeveloperScript`

`CountToolsScript.formatted_usage`

Inherited from `developerscripttools.DeveloperScript`

CountToolsScript.**formatted_version**

Inherited from `developerscriptttools.DeveloperScript`

CountToolsScript.**long_description**

CountToolsScript.**program_name**

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

CountToolsScript.**scripting_group**

CountToolsScript.**short_description**

CountToolsScript.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

CountToolsScript.**version**

Methods

CountToolsScript.**process_args** (*args*)

CountToolsScript.**setup_argument_parser** (*parser*)

Special Methods

CountToolsScript.**__call__** (*args=None*)

Inherited from `developerscriptttools.DeveloperScript`

CountToolsScript.**__eq__** (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

CountToolsScript.**__ge__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

CountToolsScript.**__gt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

CountToolsScript.**__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

CountToolsScript.**__lt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`CountToolsScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

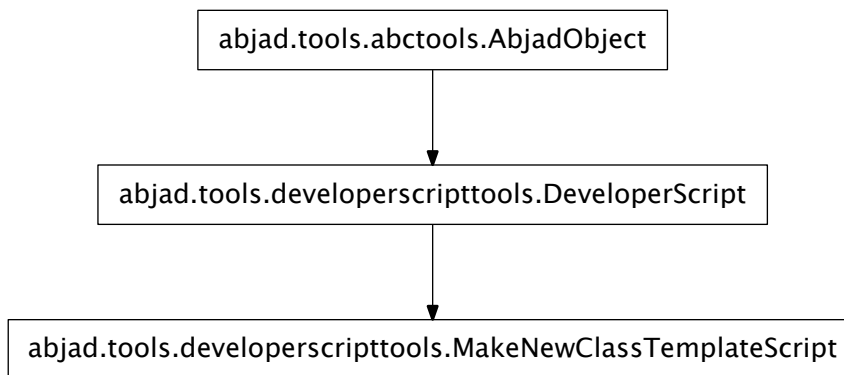
`CountToolsScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

developerscripttools.MakeNewClassTemplateScript



class developerscripttools.MakeNewClassTemplateScript

Create class stubs, complete with test subdirectory:

```
bash$ ajv new class -h
```

```
usage: make-new-class-template [-h] [--version] (-X | -M) name
```

Make a new class template file.

positional arguments:

name tools package qualified class name

optional arguments:

```
-h, --help            show this help message and exit
--version             show program's version number and exit
-X, --abjad.tools    use the Abjad abjad.tools path
-M, --mainline       use the Abjad mainline tools path
```

Return *MakeNewClassTemplateScript* instance.

Read-only Properties

`MakeNewClassTemplateScript.alias`

`MakeNewClassTemplateScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscriptttools.DeveloperScript`

`MakeNewClassTemplateScript.formatted_help`

Inherited from `developerscriptttools.DeveloperScript`

`MakeNewClassTemplateScript.formatted_usage`

Inherited from `developerscriptttools.DeveloperScript`

`MakeNewClassTemplateScript.formatted_version`

Inherited from `developerscriptttools.DeveloperScript`

`MakeNewClassTemplateScript.long_description`

`MakeNewClassTemplateScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

`MakeNewClassTemplateScript.scripting_group`

`MakeNewClassTemplateScript.short_description`

`MakeNewClassTemplateScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`MakeNewClassTemplateScript.version`

Methods

`MakeNewClassTemplateScript.process_args` (*args*)

`MakeNewClassTemplateScript.setup_argument_parser` (*parser*)

Special Methods

`MakeNewClassTemplateScript.__call__` (*args=None*)

Inherited from `developerscriptttools.DeveloperScript`

`MakeNewClassTemplateScript.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MakeNewClassTemplateScript.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MakeNewClassTemplateScript.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MakeNewClassTemplateScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MakeNewClassTemplateScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MakeNewClassTemplateScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

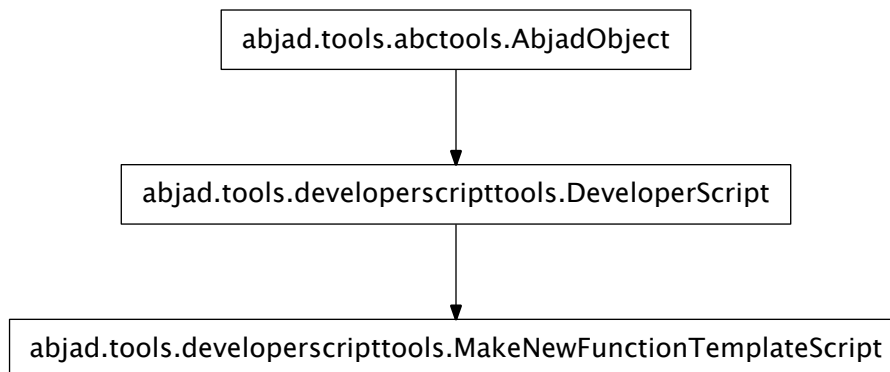
`MakeNewClassTemplateScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

developerscripttools.MakeNewFunctionTemplateScript



class `developerscripttools.MakeNewFunctionTemplateScript`

Create function stub files:

```
bash$ ajv new function -h
```

```
usage: make-new-function-template [-h] [--version] (-X | -M) name
```

Make a new function template file.

positional arguments:

name tools package qualified function name

optional arguments:

-h, --help show this help message and exit
--version show program's version number and exit
-X, --abjad.tools use the Abjad abjad.tools path
-M, --mainline use the Abjad mainline tools path

Return *MakeNewFunctionTemplateScript* instance.

Read-only Properties

MakeNewFunctionTemplateScript.**alias**

MakeNewFunctionTemplateScript.**argument_parser**

The script's instance of *argparse.ArgumentParser*.

Inherited from *developerscriptttools.DeveloperScript*

MakeNewFunctionTemplateScript.**formatted_help**

Inherited from *developerscriptttools.DeveloperScript*

MakeNewFunctionTemplateScript.**formatted_usage**

Inherited from *developerscriptttools.DeveloperScript*

MakeNewFunctionTemplateScript.**formatted_version**

Inherited from *developerscriptttools.DeveloperScript*

MakeNewFunctionTemplateScript.**long_description**

MakeNewFunctionTemplateScript.**program_name**

The name of the script, callable from the command line.

Inherited from *developerscriptttools.DeveloperScript*

MakeNewFunctionTemplateScript.**scripting_group**

MakeNewFunctionTemplateScript.**short_description**

MakeNewFunctionTemplateScript.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from *abctools.AbjadObject*

MakeNewFunctionTemplateScript.**version**

Methods

MakeNewFunctionTemplateScript.**process_args**(args)

MakeNewFunctionTemplateScript.**setup_argument_parser**(parser)

Special Methods

MakeNewFunctionTemplateScript.**__call__**(args=None)

Inherited from *developerscriptttools.DeveloperScript*

MakeNewFunctionTemplateScript.**__eq__**(arg)

True when *id(self)* equals *id(arg)*.

Return boolean.

Inherited from *abctools.AbjadObject*

`MakeNewFunctionTemplateScript.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MakeNewFunctionTemplateScript.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MakeNewFunctionTemplateScript.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MakeNewFunctionTemplateScript.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MakeNewFunctionTemplateScript.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

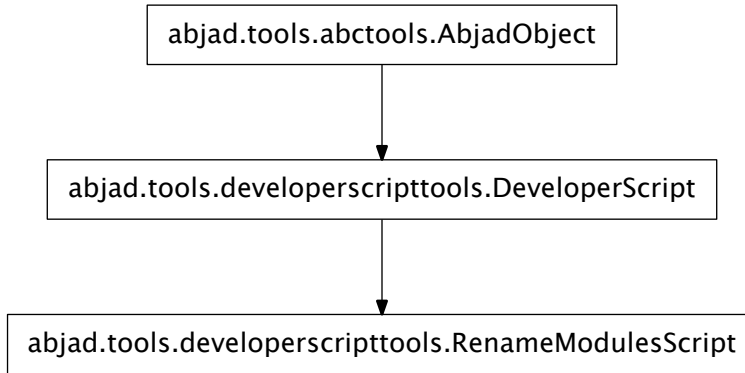
Inherited from `abctools.AbjadObject`

`MakeNewFunctionTemplateScript.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

developerscripttools.RenameModulesScript



class `developerscripttools.RenameModulesScript`

Rename classes and functions.

Handle renaming the module and package, as well as any tests, documentation or mentions of the class throughout the Abjad codebase:

```
$ ajv rename -h
usage: rename-modules [-h] [--version] (-C | -F)
```

Rename public modules.

optional arguments:

```
-h, --help          show this help message and exit
--version           show program's version number and exit
-C, --classes       rename classes
-F, --functions     rename functions
```

Return *RenameModulesScript* instance.

Read-only Properties

`RenameModulesScript.alias`

`RenameModulesScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscripttools.DeveloperScript`

`RenameModulesScript.formatted_help`

Inherited from `developerscripttools.DeveloperScript`

`RenameModulesScript.formatted_usage`

Inherited from `developerscripttools.DeveloperScript`

`RenameModulesScript.formatted_version`

Inherited from `developerscripttools.DeveloperScript`

`RenameModulesScript.long_description`

`RenameModulesScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscripttools.DeveloperScript`

`RenameModulesScript.scripting_group`

`RenameModulesScript.short_description`

`RenameModulesScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`RenameModulesScript.version`

Methods

`RenameModulesScript.process_args(args)`

`RenameModulesScript.setup_argument_parser(parser)`

Special Methods

`RenameModulesScript.__call__(args=None)`

Inherited from `developerscripttools.DeveloperScript`

`RenameModulesScript.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`RenameModulesScript.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RenameModulesScript.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`RenameModulesScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RenameModulesScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`RenameModulesScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

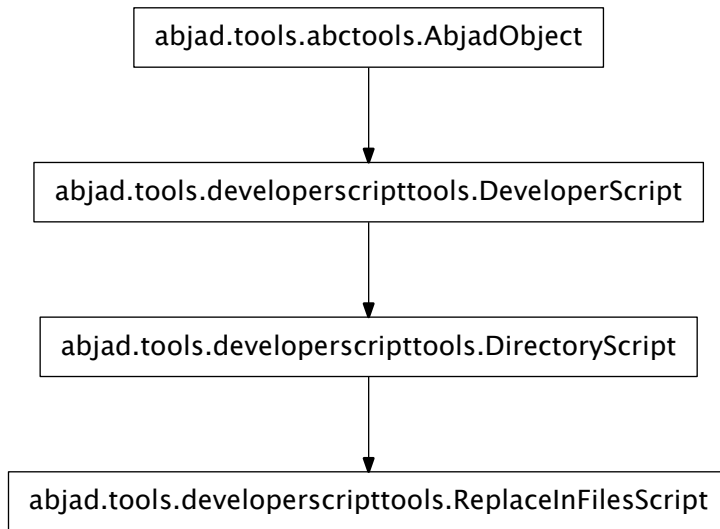
`RenameModulesScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`developerscripttools.ReplaceInFilesScript`



`class developerscripttools.ReplaceInFilesScript`

Replace text in files recursively:

```

bash$ ajv replace text -h
usage: replace-in-files [-h] [--version] [--verbose] [-Y] [-R] [-W]
                        [-F PATTERN] [-D PATTERN]
                        path old new
  
```

Replace text.

positional arguments:

<code>path</code>	directory tree to be recursed over
<code>old</code>	old text
<code>new</code>	new text

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--version</code>	show program's version number and exit
<code>--verbose</code>	print replacement info even when <code>--force</code> flag is set.
<code>-Y, --force</code>	force "yes" to every replacement
<code>-R, --regex</code>	treat "old" as a regular expression


```
-W, --whole-words-only          match only whole words, similar to grep's "-w" flag
-F PATTERN, --without-files PATTERN
                                Exclude files matching pattern(s)
-D PATTERN, --without-dirs PATTERN
                                Exclude folders matching pattern(s)
```

Multiple patterns for excluding files or folders can be specified by restating the `--without-files` or `--without-dirs` commands:

```
bash$ ajv replace text . foo bar -F *.txt -F *.rst -F *.htm
```

Return *ReplaceInFilesScript* instance.

Read-only Properties

`ReplaceInFilesScript.alias`

`ReplaceInFilesScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscripttools.DeveloperScript`

`ReplaceInFilesScript.formatted_help`

Inherited from `developerscripttools.DeveloperScript`

`ReplaceInFilesScript.formatted_usage`

Inherited from `developerscripttools.DeveloperScript`

`ReplaceInFilesScript.formatted_version`

Inherited from `developerscripttools.DeveloperScript`

`ReplaceInFilesScript.long_description`

`ReplaceInFilesScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscripttools.DeveloperScript`

`ReplaceInFilesScript.scripting_group`

`ReplaceInFilesScript.short_description`

`ReplaceInFilesScript.skipped_directories`

`ReplaceInFilesScript.skipped_files`

`ReplaceInFilesScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`ReplaceInFilesScript.version`

Methods

`ReplaceInFilesScript.process_args(args)`

`ReplaceInFilesScript.setup_argument_parser(parser)`

Special Methods

`ReplaceInFilesScript.__call__(args=None)`

Inherited from `developerscripttools.DeveloperScript`

`ReplaceInFilesScript.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ReplaceInFilesScript.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReplaceInFilesScript.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ReplaceInFilesScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReplaceInFilesScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReplaceInFilesScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

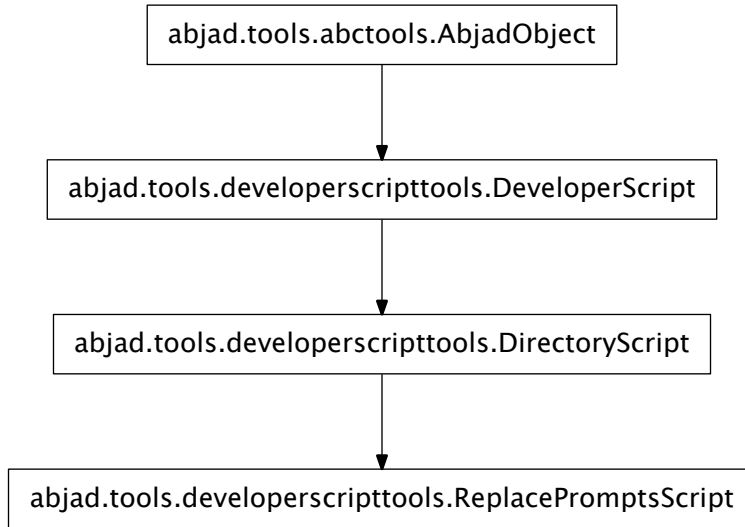
Inherited from `abctools.AbjadObject`

`ReplaceInFilesScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

developerscripttools.ReplacePromptsScript**class** `developerscripttools.ReplacePromptsScript`

Replace prompts in code examples recursively:

```
bash$ ajv replace prompts -h
usage: replace-prompts [-h] [--version] [-PA | -AP] path
```

Replace prompts.

positional arguments:

path directory tree to be recursed over

optional arguments:

```
-h, --help                      show this help message and exit
--version                      show program's version number and exit
-PA, --python-to-abjad        replace Python (">>>") prompts with Abjad ("abjad>")
                               prompts
-AP, --abjad-to-python        replace Abjad ("abjad>") prompts with Python (">>>")
                               prompts
```

examples:

Replace Python prompts with Abjad prompts in the current directory:

```
$ abj-dev replace prompts --python-to-abjad .
```

Replace Abjad prompts with Python prompts in the grandparent directory:

```
$ abj-dev replace prompts --abjad-to-python ../../..
```

ReplacePromptsScript uses *ReplaceInFilesScript* for its replacement functionality.

Return *ReplacePromptsScript* instance.

Read-only Properties

`ReplacePromptsScript.alias`

`ReplacePromptsScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscriptttools.DeveloperScript`

`ReplacePromptsScript.formatted_help`

Inherited from `developerscriptttools.DeveloperScript`

`ReplacePromptsScript.formatted_usage`

Inherited from `developerscriptttools.DeveloperScript`

`ReplacePromptsScript.formatted_version`

Inherited from `developerscriptttools.DeveloperScript`

`ReplacePromptsScript.long_description`

`ReplacePromptsScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

`ReplacePromptsScript.scripting_group`

`ReplacePromptsScript.short_description`

`ReplacePromptsScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`ReplacePromptsScript.version`

Methods

`ReplacePromptsScript.process_args` (*args*)

`ReplacePromptsScript.setup_argument_parser` (*parser*)

Special Methods

`ReplacePromptsScript.__call__` (*args=None*)

Inherited from `developerscriptttools.DeveloperScript`

`ReplacePromptsScript.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ReplacePromptsScript.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReplacePromptsScript.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ReplacePromptsScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReplacePromptsScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReplacePromptsScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

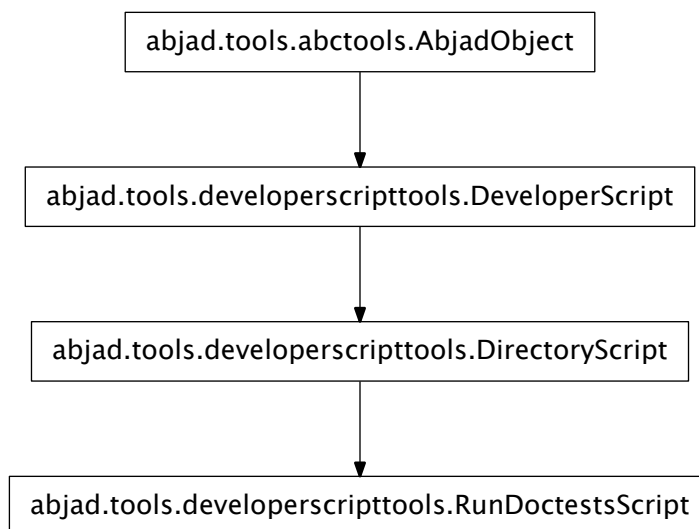
`ReplacePromptsScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

developerscripttools.RunDoctestsScript



class `developerscripttools.RunDoctestsScript`

Run doctests on all Python files in current directory recursively:

```
bash$ ajv doctest -h
usage: run-doctests [-h] [--version]
```

Run doctests on all modules in current path.

optional arguments:

```
-h, --help  show this help message and exit
--version  show program's version number and exit
```

Return *RunDoctestsScript* instance.

Read-only Properties

`RunDoctestsScript.alias`

`RunDoctestsScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscripttools.DeveloperScript`

`RunDoctestsScript.formatted_help`

Inherited from `developerscripttools.DeveloperScript`

`RunDoctestsScript.formatted_usage`

Inherited from `developerscripttools.DeveloperScript`

`RunDoctestsScript.formatted_version`

Inherited from `developerscripttools.DeveloperScript`

`RunDoctestsScript.long_description`

`RunDoctestsScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscripttools.DeveloperScript`

`RunDoctestsScript.scripting_group`

`RunDoctestsScript.short_description`

`RunDoctestsScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`RunDoctestsScript.version`

Methods

`RunDoctestsScript.process_args` (*args*)

`RunDoctestsScript.setup_argument_parser` (*parser*)

Special Methods

`RunDoctestsScript.__call__` (*args=None*)

Inherited from `developerscripttools.DeveloperScript`

`RunDoctestsScript.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`RunDoctestsScript.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`RunDoctestsScript.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

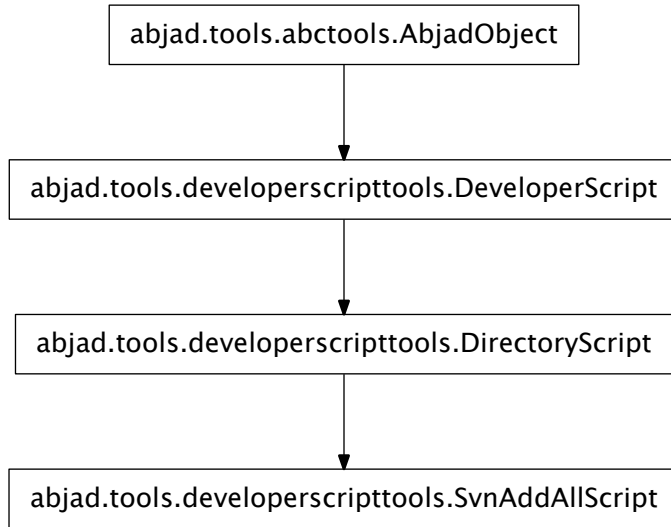
`RunDoctestsScript.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`RunDoctestsScript.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`RunDoctestsScript.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`RunDoctestsScript.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.
Inherited from `abctools.AbjadObject`

developerscripttools.SvnAddAllScript



class `developerscripttools.SvnAddAllScript`

Run *svn add* on all unversioned files in path:

```

bash$ ajv svn add -h
usage: svn-add-all [-h] [--version] path

"svn add" all unversioned files in PATH.

positional arguments:
  path                directory tree to be recursed over

optional arguments:
  -h, --help          show this help message and exit
  --version            show program's version number and exit
  
```

Return *SvnAddAllScript* instance.

Read-only Properties

`SvnAddAllScript.alias`

`SvnAddAllScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscripttools.DeveloperScript`

`SvnAddAllScript.formatted_help`

Inherited from `developerscripttools.DeveloperScript`

`SvnAddAllScript.formatted_usage`

Inherited from `developerscripttools.DeveloperScript`

`SvnAddAllScript.formatted_version`

Inherited from `developerscripttools.DeveloperScript`

`SvnAddAllScript.long_description`

`SvnAddAllScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscripttools.DeveloperScript`

`SvnAddAllScript.scripting_group`

`SvnAddAllScript.short_description`

`SvnAddAllScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`SvnAddAllScript.version`

Methods

`SvnAddAllScript.process_args` (*args*)

`SvnAddAllScript.setup_argument_parser` (*parser*)

Special Methods

`SvnAddAllScript.__call__` (*args=None*)

Inherited from `developerscripttools.DeveloperScript`

`SvnAddAllScript.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SvnAddAllScript.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnAddAllScript.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SvnAddAllScript.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnAddAllScript.__lt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnAddAllScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

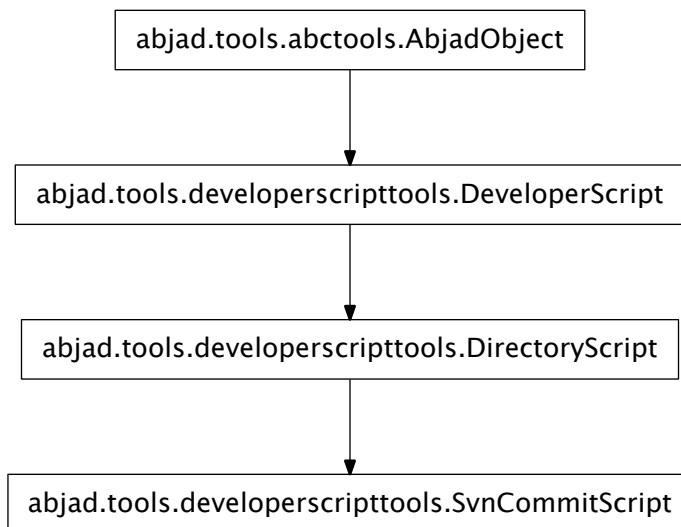
`SvnAddAllScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`developerscripttools.SvnCommitScript`



class `developerscripttools.SvnCommitScript`

Run *svn commit*, using the commit message stored in the *.abjad* directory.

The commit message will be printed to the terminal, and must be manually accepted or rejected before proceeding:

```
bash$ ajv svn ci -h
usage: svn-commit [-h] [--version] path
```

"svn commit", using previously written commit message.

positional arguments:

path commit the path PATH

optional arguments:

-h, --help show this help message and exit
 --version show program's version number and exit

Return *SvnCommitScript* instance.

Read-only Properties

`SvnCommitScript.alias`

`SvnCommitScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscriptttools.DeveloperScript`

`SvnCommitScript.formatted_help`

Inherited from `developerscriptttools.DeveloperScript`

`SvnCommitScript.formatted_usage`

Inherited from `developerscriptttools.DeveloperScript`

`SvnCommitScript.formatted_version`

Inherited from `developerscriptttools.DeveloperScript`

`SvnCommitScript.long_description`

`SvnCommitScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

`SvnCommitScript.scripting_group`

`SvnCommitScript.short_description`

`SvnCommitScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`SvnCommitScript.version`

Methods

`SvnCommitScript.process_args(args)`

`SvnCommitScript.setup_argument_parser(parser)`

Special Methods

`SvnCommitScript.__call__(args=None)`

Inherited from `developerscriptttools.DeveloperScript`

`SvnCommitScript.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SvnCommitScript.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnCommitScript.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SvnCommitScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnCommitScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnCommitScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

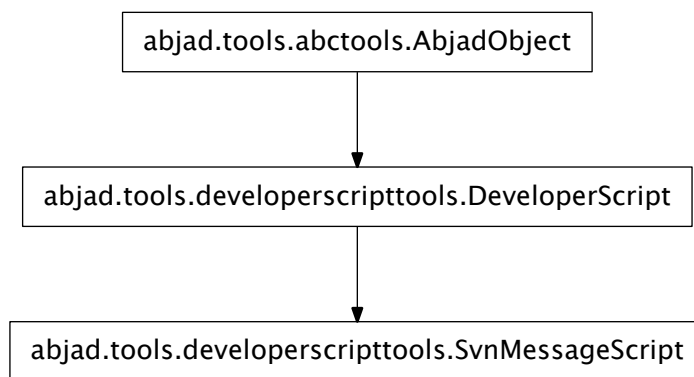
`SvnCommitScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`developerscripttools.SvnMessageScript`



class `developerscripttools.SvnMessageScript`

Edit a temporary *svn* commit message, stored in the *.abjad* directory:

```
bash$ ajv svn msg -h
usage: svn-message [-h] [--version] [-C]
```

Write commit message for future commit usage.

optional arguments:

```
-h, --help    show this help message and exit
--version     show program's version number and exit
-C, --clean   delete previous commit message before editing
```

Return *SvnMessageScript* instance.

Read-only Properties

`SvnMessageScript.alias`

`SvnMessageScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscriptttools.DeveloperScript`

`SvnMessageScript.commit_message_path`

`SvnMessageScript.formatted_help`

Inherited from `developerscriptttools.DeveloperScript`

`SvnMessageScript.formatted_usage`

Inherited from `developerscriptttools.DeveloperScript`

`SvnMessageScript.formatted_version`

Inherited from `developerscriptttools.DeveloperScript`

`SvnMessageScript.long_description`

`SvnMessageScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

`SvnMessageScript.scripting_group`

`SvnMessageScript.short_description`

`SvnMessageScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`SvnMessageScript.version`

Methods

`SvnMessageScript.process_args` (*args*)

`SvnMessageScript.setup_argument_parser` (*parser*)

Special Methods

`SvnMessageScript.__call__` (*args=None*)

Inherited from `developerscriptttools.DeveloperScript`

`SvnMessageScript.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SvnMessageScript.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnMessageScript.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SvnMessageScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnMessageScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnMessageScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

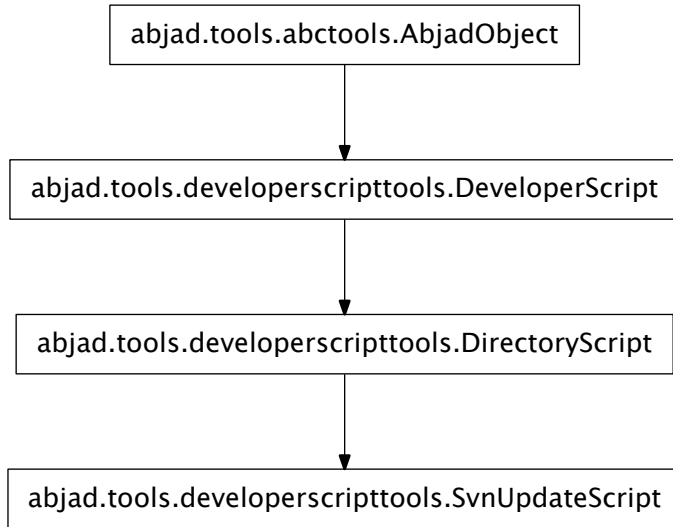
Inherited from `abctools.AbjadObject`

`SvnMessageScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

developerscripttools.SvnUpdateScript**class** `developerscripttools.SvnUpdateScript`

Run *svn up* on various Abjad paths:

```
bash$ ajv svn up -h
usage: svn-update [-h] [--version] [-C] [-P PATH | -E | -M | -R]
```

"svn update" various paths.

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--version</code>	show program's version number and exit
<code>-C, --clean</code>	remove .pyc files and __pycache__ directories before updating
<code>-P PATH, --path PATH</code>	update the path PATH
<code>-E, --abjad.tools</code>	update Abjad abjad.tools directory
<code>-M, --mainline</code>	update Abjad mainline directory
<code>-R, --root</code>	update Abjad root directory

If no path flag is specified, the current directory will be updated.

It is usually most useful to run the script with the `-clean` flag, in case there are incoming deletes, as *svn* will not delete directories containing unversioned files, such as .pyc:

```
bash$ ajv svn up -C -R
```

Return *SvnUpdateScript* instance.

Read-only Properties

`SvnUpdateScript.alias`

`SvnUpdateScript.argument_parser`

The script's instance of `argparse.ArgumentParser`.

Inherited from `developerscriptttools.DeveloperScript`

`SvnUpdateScript.formatted_help`

Inherited from `developerscriptttools.DeveloperScript`

`SvnUpdateScript.formatted_usage`

Inherited from `developerscriptttools.DeveloperScript`

`SvnUpdateScript.formatted_version`

Inherited from `developerscriptttools.DeveloperScript`

`SvnUpdateScript.long_description`

`SvnUpdateScript.program_name`

The name of the script, callable from the command line.

Inherited from `developerscriptttools.DeveloperScript`

`SvnUpdateScript.scripting_group`

`SvnUpdateScript.short_description`

`SvnUpdateScript.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`SvnUpdateScript.version`

Methods

`SvnUpdateScript.process_args(args)`

`SvnUpdateScript.setup_argument_parser(parser)`

Special Methods

`SvnUpdateScript.__call__(args=None)`

Inherited from `developerscriptttools.DeveloperScript`

`SvnUpdateScript.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SvnUpdateScript.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnUpdateScript.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SvnUpdateScript.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnUpdateScript.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SvnUpdateScript.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SvnUpdateScript.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

functions

`developerscripttools.get_developer_script_classes`

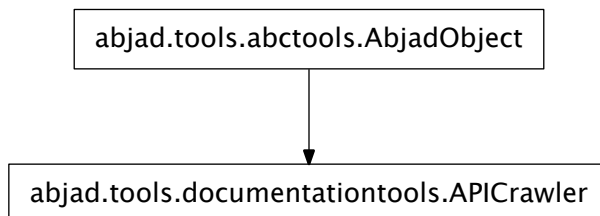
`developerscripttools.get_developer_script_classes()`

Return a list of all developer script classes.

`documentationtools`

concrete classes

`documentationtools.APICrawler`



```
class documentationtools.APICrawler (code_root, docs_root, root_package_name,
                                     ignored_directories=['test', '.svn', '__pycache__'], pre-
                                     fix='abjad.tools.')
```

Generates directories containing ReST to parallel directories containing code.

Read-only Properties

`APICrawler.code_root`

`APICrawler.docs_root`

`APICrawler.module_crawler`

`APICrawler.prefix`

`APICrawler.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`APICrawler.__call__()`

Crawl *code_root* and generate corresponding ReST in *docs_root* while ignoring ignored directories.

`APICrawler.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`APICrawler.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`APICrawler.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`APICrawler.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`APICrawler.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`APICrawler.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

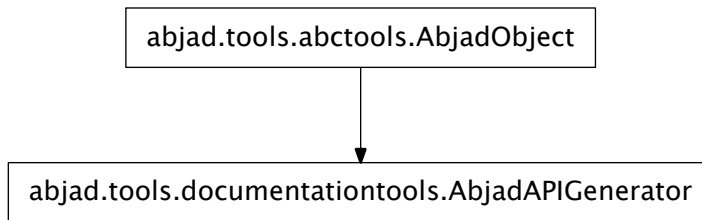
`APICrawler.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`documentationtools.AbjadAPIGenerator`



class `documentationtools.AbjadAPIGenerator`

Creates Abjad's API ReST:

- writes ReST pages for individual classes and functions
- writes the API index ReST
- handles sorting tools packages into composition, manual-loading and unstable
- handles ignoring private tools packages

Returns *AbjadAPIGenerator* instance.

Read-only Properties

`AbjadAPIGenerator.code_tools_path`

Path to Abjad tools package.

`AbjadAPIGenerator.docs_api_index_path`

Path to index.rst for Abjad API.

`AbjadAPIGenerator.docs_tools_path`

Path to tools directory inside docs.

`AbjadAPIGenerator.package_prefix`

`AbjadAPIGenerator.root_package`

`AbjadAPIGenerator.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

`AbjadAPIGenerator.tools_package_path_index`

Special Methods

AbjadAPIGenerator.__call__(verbose=False)

AbjadAPIGenerator.__eq__(arg)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__ne__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

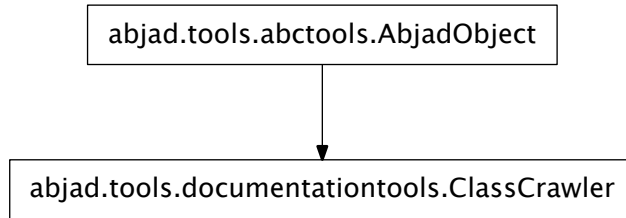
AbjadAPIGenerator.__repr__()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

documentationtools.ClassCrawler



```
class documentationtools.ClassCrawler (code_root,                include_private_objects=False,
                                       root_package_name=None)
```

Read-only Properties

`ClassCrawler.code_root`

`ClassCrawler.include_private_objects`

`ClassCrawler.module_crawler`

`ClassCrawler.root_package_name`

`ClassCrawler.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`ClassCrawler.__call__()`

`ClassCrawler.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ClassCrawler.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ClassCrawler.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ClassCrawler.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ClassCrawler.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ClassCrawler.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

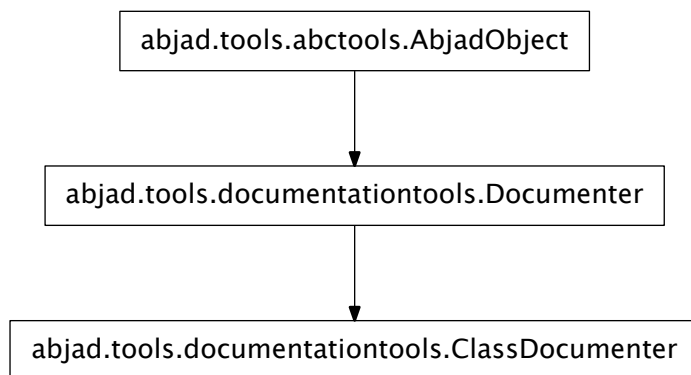
`ClassCrawler.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

documentationtools.ClassDocumenter



class `documentationtools.ClassDocumenter` (*obj*, *prefix*='abjad.tools.')

`ClassDocumenter` generates an ReST API entry for a given class:

```
>>> from abjad.tools import notetools
>>> from abjad.tools.documentationtools import ClassDocumenter

>>> documenter = ClassDocumenter(notetools.Note)
>>> rest = documenter()
```

Returns `ClassDocumenter` instance.

Read-only Properties

`ClassDocumenter.data`
`ClassDocumenter.inherited_attributes`
`ClassDocumenter.is_abstract`
`ClassDocumenter.methods`
`ClassDocumenter.module_name`
 Inherited from `documentationtools.Documenter`
`ClassDocumenter.object`
 Inherited from `documentationtools.Documenter`
`ClassDocumenter.prefix`
 Inherited from `documentationtools.Documenter`
`ClassDocumenter.readonly_properties`
`ClassDocumenter.readwrite_properties`
`ClassDocumenter.special_methods`
`ClassDocumenter.storage_format`
 Storage format of Abjad object.
 Return string.
 Inherited from `abctools.AbjadObject`

Special Methods

`ClassDocumenter.__call__()`
 Generate documentation.
 Returns string.
`ClassDocumenter.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`
`ClassDocumenter.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`
`ClassDocumenter.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`
`ClassDocumenter.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`ClassDocumenter.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ClassDocumenter.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

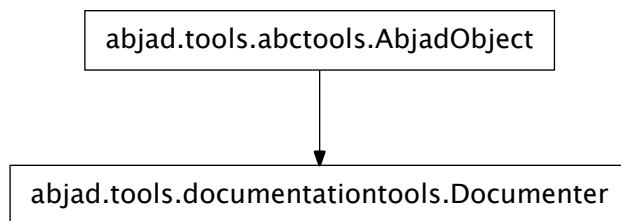
`ClassDocumenter.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

documentationtools.Documenter



class `documentationtools.Documenter` (*obj*, *prefix*='abjad.tools.')

Documenter is an abstract base class for documentation classes.

Read-only Properties

`Documenter.module_name`

`Documenter.object`

`Documenter.prefix`

`Documenter.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`Documenter.__call__()`

`Documenter.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Documenter.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Documenter.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Documenter.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Documenter.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Documenter.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

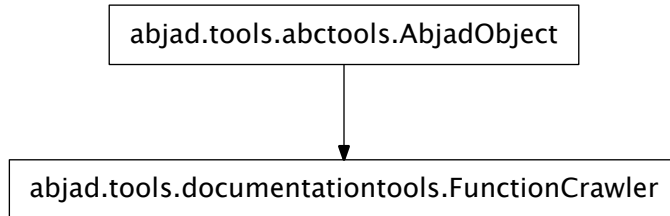
`Documenter.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

documentationtools.FunctionCrawler



```
class documentationtools.FunctionCrawler (code_root,          include_private_objects=False,
                                           root_package_name=None)
```

Read-only Properties

`FunctionCrawler.code_root`

`FunctionCrawler.include_private_objects`

`FunctionCrawler.module_crawler`

`FunctionCrawler.root_package_name`

`FunctionCrawler.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`FunctionCrawler.__call__()`

`FunctionCrawler.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`FunctionCrawler.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FunctionCrawler.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`FunctionCrawler.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FunctionCrawler.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FunctionCrawler.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

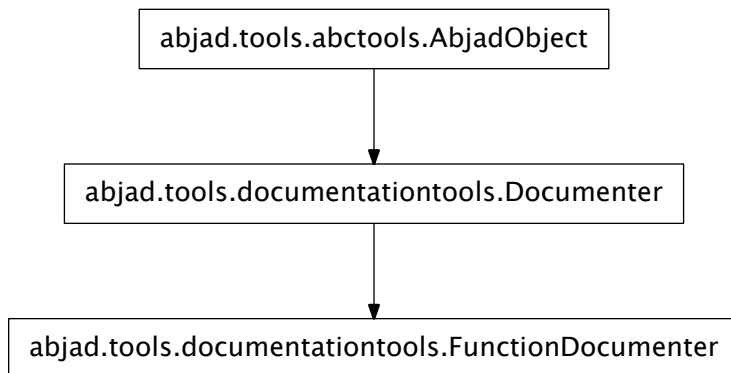
`FunctionCrawler.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`documentationtools.FunctionDocumenter`



class `documentationtools.FunctionDocumenter` (*obj*, *prefix*='abjad.tools.')

FunctionDocumenter generates an ReST entry for a given function:

```

>>> from abjad.tools.documentationtools import *

>>> from abjad.tools.notetools import make_notes
>>> documenter = FunctionDocumenter(make_notes)
>>> print documenter()
notetools.make_notes
=====

.. autofunction:: abjad.tools.notetools.make_notes.make_notes
  
```

Returns `FunctionDocumenter` instance.

Read-only Properties

`FunctionDocumenter.module_name`

Inherited from `documentationtools.Documenter`

`FunctionDocumenter.object`

Inherited from `documentationtools.Documenter`

`FunctionDocumenter.prefix`

Inherited from `documentationtools.Documenter`

`FunctionDocumenter.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`FunctionDocumenter.__call__()`

Generate documentation.

Returns string.

`FunctionDocumenter.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

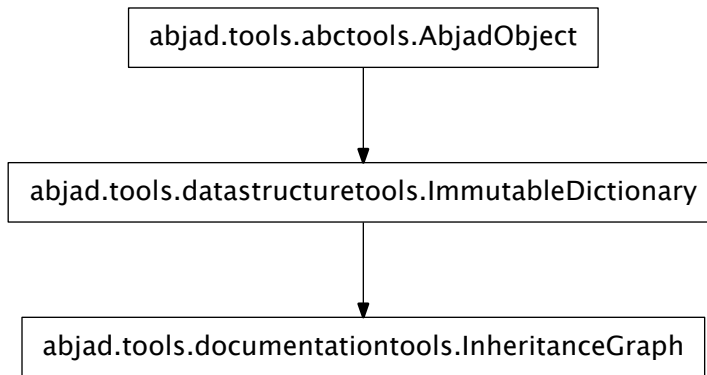
`FunctionDocumenter.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`documentationtools.InheritanceGraph`



class `documentationtools.InheritanceGraph(*args, **kwargs)`

Generates a graph of a class or collection of classes as a dictionary of parent-children relationships:

```

>>> from abjad.tools.documentationtools import InheritanceGraph

>>> class A(object): pass
...
>>> class B(A): pass
...
>>> class C(B): pass
...
>>> class D(B): pass
...
>>> class E(C, D): pass
...
>>> class F(A): pass
...

>>> graph = InheritanceGraph(F, E)

>>> result = sorted(graph.items(), key=lambda x: x[0].__name__)
>>> for parent, children in result:
...     parent, tuple(sorted(children, key=lambda x: x.__name__))
(<class '__main__.A'>, (<class '__main__.B'>, <class '__main__.F'>))
(<class '__main__.B'>, (<class '__main__.C'>, <class '__main__.D'>))
(<class '__main__.C'>, (<class '__main__.E'>,))
(<class '__main__.D'>, (<class '__main__.E'>,))
(<class '__main__.E'>, ())
  
```

```
(<class '__main__.F'>, ())
(<type 'object'>, (<class '__main__.A'>,))
```

InheritanceGraph may be instantiated from one or more instances, classes or modules. If instantiated from a module, all public classes in that module will be taken into the graph.

A *root_class* keyword may be defined at instantiation, which filters out all classes from the graph which do not inherit from that *root_class* (or are not already the *root_class*):

```
>>> graph = InheritanceGraph(A, B, C, D, E, F, root_class=B)
>>> for parent, children in sorted(graph.items()):
...     parent, tuple(sorted(children))
(<class '__main__.B'>, (<class '__main__.C'>, <class '__main__.D'>))
(<class '__main__.C'>, (<class '__main__.E'>,))
(<class '__main__.D'>, (<class '__main__.E'>,))
(<class '__main__.E'>, ())
```

Returns InheritanceGraph instance.

Read-only Properties

InheritanceGraph.**graphviz_format**

InheritanceGraph.**root_class**

InheritanceGraph.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

InheritanceGraph.**clear**() → None. Remove all items from D.

Inherited from `__builtin__.dict`

InheritanceGraph.**copy**() → a shallow copy of D

Inherited from `__builtin__.dict`

InheritanceGraph.**get**(*k*[, *d*]) → D[*k*] if *k* in D, else *d*. *d* defaults to None.

Inherited from `__builtin__.dict`

InheritanceGraph.**has_key**(*k*) → True if D has a key *k*, else False

Inherited from `__builtin__.dict`

InheritanceGraph.**items**() → list of D's (key, value) pairs, as 2-tuples

Inherited from `__builtin__.dict`

InheritanceGraph.**iteritems**() → an iterator over the (key, value) items of D

Inherited from `__builtin__.dict`

InheritanceGraph.**iterkeys**() → an iterator over the keys of D

Inherited from `__builtin__.dict`

InheritanceGraph.**itervalues**() → an iterator over the values of D

Inherited from `__builtin__.dict`

InheritanceGraph.**keys**() → list of D's keys

Inherited from `__builtin__.dict`

InheritanceGraph.**pop**(*k*[, *d*]) → *v*, remove specified key and return the corresponding value.

If key is not found, *d* is returned if given, otherwise `KeyError` is raised

Inherited from `__builtin__.dict`

`InheritanceGraph.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Inherited from `__builtin__.dict`

`InheritanceGraph.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Inherited from `__builtin__.dict`

`InheritanceGraph.update([E], **F)` → None. Update D from dict/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Inherited from `__builtin__.dict`

`InheritanceGraph.values()` → list of D's values

Inherited from `__builtin__.dict`

`InheritanceGraph.viewitems()` → a set-like object providing a view on D's items

Inherited from `__builtin__.dict`

`InheritanceGraph.viewkeys()` → a set-like object providing a view on D's keys

Inherited from `__builtin__.dict`

`InheritanceGraph.viewvalues()` → an object providing a view on D's values

Inherited from `__builtin__.dict`

Special Methods

`InheritanceGraph.__cmp__(y)` <==> *cmp(x, y)*

Inherited from `__builtin__.dict`

`InheritanceGraph.__contains__(k)` → True if D has a key k, else False

Inherited from `__builtin__.dict`

`InheritanceGraph.__delitem__(*args)`

Inherited from `datastructuretools.ImmutableDictionary`

`InheritanceGraph.__eq__()`

x.__eq__(y) <==> *x==y*

Inherited from `__builtin__.dict`

`InheritanceGraph.__ge__()`

x.__ge__(y) <==> *x>=y*

Inherited from `__builtin__.dict`

`InheritanceGraph.__getitem__()`

x.__getitem__(y) <==> *x[y]*

Inherited from `__builtin__.dict`

`InheritanceGraph.__gt__()`

x.__gt__(y) <==> *x>y*

Inherited from `__builtin__.dict`

`InheritanceGraph.__iter__()` <==> *iter(x)*

Inherited from `__builtin__.dict`

`InheritanceGraph.__le__()`

x.__le__(y) <==> *x<=y*

Inherited from `__builtin__.dict`

```
InheritanceGraph.__len__() <==> len(x)
    Inherited from __builtin__.dict

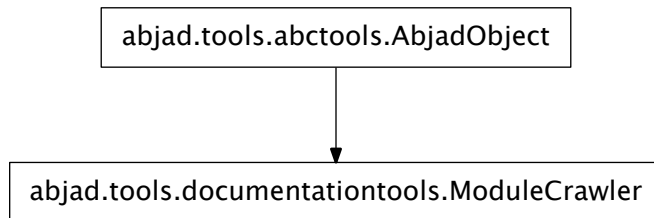
InheritanceGraph.__lt__()
    x.__lt__(y) <==> x<y
    Inherited from __builtin__.dict

InheritanceGraph.__ne__()
    x.__ne__(y) <==> x!=y
    Inherited from __builtin__.dict

InheritanceGraph.__repr__() <==> repr(x)
    Inherited from __builtin__.dict

InheritanceGraph.__setitem__(*args)
    Inherited from datastructuretools.ImmutableDictionary
```

documentationtools.ModuleCrawler



```
class documentationtools.ModuleCrawler(code_root='', ignored_directories=['test', 'svn',
                                                                    '__pycache__'], root_package_name=None)
    Crawls code_root, yielding all module objects whose name begins with root_package_name.
    Return ModuleCrawler instance.
```

Read-only Properties

```
ModuleCrawler.code_root
ModuleCrawler.ignored_directories
ModuleCrawler.root_package_name
ModuleCrawler.storage_format
    Storage format of Abjad object.
    Return string.
    Inherited from abctools.AbjadObject
```

Special Methods

`ModuleCrawler.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`ModuleCrawler.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`ModuleCrawler.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`ModuleCrawler.__iter__()`

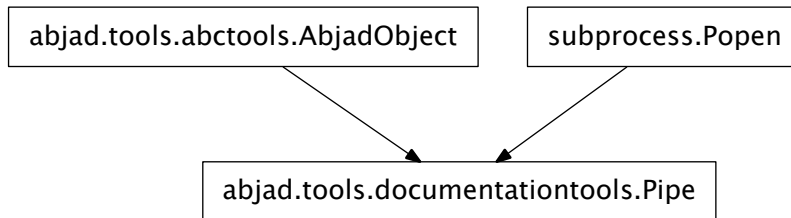
`ModuleCrawler.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`ModuleCrawler.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`ModuleCrawler.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`ModuleCrawler.__repr__()`
 Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
 Return string.
 Inherited from `abctools.AbjadObject`

documentationtools.Pipe



class documentationtools.**Pipe** (*executable='python', arguments=['-i'], timeout=0*)
 A two-way, non-blocking pipe for interprocess communication:

```

>>> from abjad.tools.documentationtools import Pipe

>>> pipe = Pipe('python', ['-i'])
>>> pipe.writeline('my_list = [1, 2, 3]')
>>> pipe.writeline('print my_list')
  
```

Return *Pipe* instance.

Read-only Properties

Pipe.**arguments**

Pipe.**executable**

Pipe.**storage_format**

Storage format of Abjad object.

Return string.

Inherited from abctools.AbjadObject

Pipe.**timeout**

Methods

Pipe.**close()**

Close the pipe.

Pipe.**communicate** (*input=None*)

Interact with process: Send data to stdin. Read data from stdout and stderr, until end-of-file is reached. Wait for process to terminate. The optional input argument should be a string to be sent to the child process, or None, if no data should be sent to the child.

communicate() returns a tuple (stdout, stderr).

Inherited from subprocess.Popen

Pipe.**kill()**

Kill the process with SIGKILL

Inherited from subprocess.Popen

`Pipe.poll()`
Inherited from `subprocess.Popen`

`Pipe.read()`
Read from the pipe.

`Pipe.read_wait(seconds=0.01)`
Try to read from the pipe. Wait *seconds* if nothing comes out, and repeat.
Should be used with caution, as this may loop forever.

`Pipe.send_signal(sig)`
Send a signal to the process
Inherited from `subprocess.Popen`

`Pipe.terminate()`
Terminate the process with SIGTERM
Inherited from `subprocess.Popen`

`Pipe.wait()`
Wait for child process to terminate. Returns `returncode` attribute.
Inherited from `subprocess.Popen`

`Pipe.write(data)`
Write *data* into the pipe.

`Pipe.write_line(data)`
Write *data* into the pipe, terminated by a newline.

Special Methods

`Pipe.__del__(_maxint=9223372036854775807, _active=[])`
Inherited from `subprocess.Popen`

`Pipe.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`Pipe.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`Pipe.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception
Inherited from `abctools.AbjadObject`

`Pipe.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.
Inherited from `abctools.AbjadObject`

`Pipe.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Pipe.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Pipe.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

functions

`documentationtools.make_ligeti_example_lilypond_file`

`documentationtools.make_ligeti_example_lilypond_file(music=None)`

New in version 2.9. Make Ligeti example LilyPond file.

Return LilyPond file.

`documentationtools.make_reference_manual_lilypond_file`

`documentationtools.make_reference_manual_lilypond_file(music=None)`

New in version 2.9. Make reference manual LilyPond file.

```
>>> from abjad.tools import documentationtools

>>> score = Score([Staff('c d e f')])
>>> lilypond_file = documentationtools.make_reference_manual_lilypond_file(score)

>>> f(lilypond_file)
% Abjad revision 5653
% 2012-05-19 11:21

\version "2.15.37"
\language "english"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"

\layout {
  indent = #0
  ragged-right = ##t
  \context {
    \Score
    \remove Bar_number_engraver
    \override SpacingSpanner #'strict-grace-spacing = ##t
    \override SpacingSpanner #'strict-note-spacing = ##t
    \override SpacingSpanner #'uniform-stretching = ##t
    \override TupletBracket #'bracket-visibility = ##t
```

```

\override TupletBracket #'minimum-length = #3
\override TupletBracket #'padding = #2
\override TupletBracket #'springs-and-rods = #ly:spanner::set-spacing-rods
\override TupletNumber #'text = #tuplet-number::calc-fraction-text
proportionalNotationDuration = #(ly:make-moment 1 32)
tupletFullLength = True
}
}

\score {
  \new Score <<
    \new Staff {
      c4
      d4
      e4
      f4
    }
  >>
}

```

Return LilyPond file.

documentationtools.make_text_alignment_example_lilypond_file

documentationtools.make_text_alignment_example_lilypond_file(*music=None*)

New in version 2.9. Make text-alignment example LilyPond file with *music*.

```

>>> from abjad.tools import documentationtools

>>> score = Score([Staff('c d e f')])
>>> lilypond_file = documentationtools.make_text_alignment_example_lilypond_file(score)

>>> f(lilypond_file)
% Abjad revision 5651
% 2012-05-19 10:04

\version "2.15.37"
\language "english"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"

#(set-global-staff-size 18)

\layout {
  indent = #0
  ragged-right = ##t
  \context {
    \Score
    \remove Bar_number_engraver
    \remove Default_bar_line_engraver
    \override Clef #'transparent = ##t
    \override SpacingSpanner #'strict-grace-spacing = ##t
    \override SpacingSpanner #'strict-note-spacing = ##t
    \override SpacingSpanner #'uniform-stretching = ##t
    \override TextScript #'staff-padding = #4
    proportionalNotationDuration = #(ly:make-moment 1 32)
  }
}

```

```
\paper {
  bottom-margin = #10
  left-margin = #10
  line-width = #150
  system-system-spacing = #'(
    (basic_distance . 0) (minimum_distance . 0) (padding . 15) (stretchability . 0))
}

\score {
  \new Score <<
    \new Staff {
      c4
      d4
      e4
      f4
    }
  >>
}
```

Return LilyPond file.

exceptiontools

concrete classes

exceptiontools.AssignabilityError

abjad.tools.exceptiontools.AssignabilityError

class tools.**AssignabilityError**

Duration can not be assigned to note, rest or chord.

Special Methods

AssignabilityError.__delattr__()
 x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

AssignabilityError.__getitem__()
 x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

AssignabilityError.__getslice__()
 x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

`AssignabilityError.__repr__()` $\iff repr(x)$
 Inherited from `exceptions.BaseException`

`AssignabilityError.__setattr__()`
`x.__setattr__('name', value) \iff x.name = value`
 Inherited from `exceptions.BaseException`

`AssignabilityError.__setstate__()`
 Inherited from `exceptions.BaseException`

`AssignabilityError.__str__()` $\iff str(x)$
 Inherited from `exceptions.BaseException`

`AssignabilityError.__unicode__()`
 Inherited from `exceptions.BaseException`

`exceptiontools.ClefError`

`abjad.tools.exceptiontools.ClefError`

class `tools.ClefError`
 General clef error.

Special Methods

`ClefError.__delattr__()`
`x.__delattr__('name') \iff del x.name`
 Inherited from `exceptions.BaseException`

`ClefError.__getitem__()`
`x.__getitem__(y) \iff x[y]`
 Inherited from `exceptions.BaseException`

`ClefError.__getslice__()`
`x.__getslice__(i, j) \iff x[i:j]`
 Use of negative indices is not supported.
 Inherited from `exceptions.BaseException`

`ClefError.__repr__()` $\iff repr(x)$
 Inherited from `exceptions.BaseException`

`ClefError.__setattr__()`
`x.__setattr__('name', value) \iff x.name = value`
 Inherited from `exceptions.BaseException`

`ClefError.__setstate__()`
 Inherited from `exceptions.BaseException`

`ClefError.__str__()` $\iff str(x)$
Inherited from `exceptions.BaseException`

`ClefError.__unicode__()`
Inherited from `exceptions.BaseException`

`exceptiontools.ContainmentError`

`abjad.tools.exceptiontools.ContainmentError`

class `tools.ContainmentError`

General containment error.

Special Methods

`ContainmentError.__delattr__()`
`x.__delattr__('name')` \iff `del x.name`
Inherited from `exceptions.BaseException`

`ContainmentError.__getitem__()`
`x.__getitem__(y)` \iff `x[y]`
Inherited from `exceptions.BaseException`

`ContainmentError.__getslice__()`
`x.__getslice__(i, j)` \iff `x[i:j]`
Use of negative indices is not supported.
Inherited from `exceptions.BaseException`

`ContainmentError.__repr__()` $\iff repr(x)$
Inherited from `exceptions.BaseException`

`ContainmentError.__setattr__()`
`x.__setattr__('name', value)` \iff `x.name = value`
Inherited from `exceptions.BaseException`

`ContainmentError.__setstate__()`
Inherited from `exceptions.BaseException`

`ContainmentError.__str__()` $\iff str(x)$
Inherited from `exceptions.BaseException`

`ContainmentError.__unicode__()`
Inherited from `exceptions.BaseException`

exceptiontools.ContextContainmentError

abjad.tools.exceptiontools.ContextContainmentError

class tools.ContextContainmentError

Context can not contain other context.

Special Methods

ContextContainmentError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

ContextContainmentError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

ContextContainmentError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

ContextContainmentError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

ContextContainmentError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

ContextContainmentError.__setstate__()

Inherited from exceptions.BaseException

ContextContainmentError.__str__() <==> str(x)

Inherited from exceptions.BaseException

ContextContainmentError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.ContiguityError

abjad.tools.exceptiontools.ContiguityError

class tools.**ContiguityError**

Input is not contiguous.

Special Methods

ContiguityError.__delattr__()
`x.__delattr__('name') <==> del x.name`
 Inherited from `exceptions.BaseException`

ContiguityError.__getitem__()
`x.__getitem__(y) <==> x[y]`
 Inherited from `exceptions.BaseException`

ContiguityError.__getslice__()
`x.__getslice__(i, j) <==> x[i:j]`
 Use of negative indices is not supported.
 Inherited from `exceptions.BaseException`

ContiguityError.__repr__() <==> `repr(x)`
 Inherited from `exceptions.BaseException`

ContiguityError.__setattr__()
`x.__setattr__('name', value) <==> x.name = value`
 Inherited from `exceptions.BaseException`

ContiguityError.__setstate__()
 Inherited from `exceptions.BaseException`

ContiguityError.__str__() <==> `str(x)`
 Inherited from `exceptions.BaseException`

ContiguityError.__unicode__()
 Inherited from `exceptions.BaseException`

exceptiontools.CyclicNodeError

abjad.tools.exceptiontools.CyclicNodeError

class tools.**CyclicNodeError**

Node is in cyclic relationship.

Special Methods

CyclicNodeError.__delattr__()
 x.__delattr__('name') <==> del x.name
 Inherited from exceptions.BaseException

CyclicNodeError.__getitem__()
 x.__getitem__(y) <==> x[y]
 Inherited from exceptions.BaseException

CyclicNodeError.__getslice__()
 x.__getslice__(i, j) <==> x[i:j]
 Use of negative indices is not supported.
 Inherited from exceptions.BaseException

CyclicNodeError.__repr__() <==> repr(x)
 Inherited from exceptions.BaseException

CyclicNodeError.__setattr__()
 x.__setattr__('name', value) <==> x.name = value
 Inherited from exceptions.BaseException

CyclicNodeError.__setstate__()
 Inherited from exceptions.BaseException

CyclicNodeError.__str__() <==> str(x)
 Inherited from exceptions.BaseException

CyclicNodeError.__unicode__()
 Inherited from exceptions.BaseException

exceptiontools.DurationError

abjad.tools.exceptiontools.DurationError

class tools.**DurationError**

General duration error.

Special Methods

DurationError.**__delattr__**()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

DurationError.**__getitem__**()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

DurationError.**__getslice__**()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

DurationError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

DurationError.**__setattr__**()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

DurationError.**__setstate__**()

Inherited from exceptions.BaseException

DurationError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

DurationError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.ExtraMarkError

abjad.tools.exceptiontools.ExtraMarkError

class tools.**ExtraMarkError**

More than one mark is found for single-mark operation.

Special Methods

ExtraMarkError.**__delattr__**()

`x.__delattr__('name') <==> del x.name`

Inherited from exceptions.BaseException

ExtraMarkError.**__getitem__**()

`x.__getitem__(y) <==> x[y]`

Inherited from exceptions.BaseException

ExtraMarkError.**__getslice__**()

`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from exceptions.BaseException

ExtraMarkError.**__repr__**() <==> *repr*(x)

Inherited from exceptions.BaseException

ExtraMarkError.**__setattr__**()

`x.__setattr__('name', value) <==> x.name = value`

Inherited from exceptions.BaseException

ExtraMarkError.**__setstate__**()

Inherited from exceptions.BaseException

ExtraMarkError.**__str__**() <==> *str*(x)

Inherited from exceptions.BaseException

ExtraMarkError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.ExtraNamedComponentError

abjad.tools.exceptiontools.ExtraNamedComponentError

class tools.**ExtraNamedComponentError**
 More than one component with the same name is found..

Special Methods

`ExtraNamedComponentError.__delattr__()`
`x.__delattr__('name') <==> del x.name`
 Inherited from `exceptions.BaseException`

`ExtraNamedComponentError.__getitem__()`
`x.__getitem__(y) <==> x[y]`
 Inherited from `exceptions.BaseException`

`ExtraNamedComponentError.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
 Use of negative indices is not supported.
 Inherited from `exceptions.BaseException`

`ExtraNamedComponentError.__repr__() <==> repr(x)`
 Inherited from `exceptions.BaseException`

`ExtraNamedComponentError.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`
 Inherited from `exceptions.BaseException`

`ExtraNamedComponentError.__setstate__()`
 Inherited from `exceptions.BaseException`

`ExtraNamedComponentError.__str__() <==> str(x)`
 Inherited from `exceptions.BaseException`

`ExtraNamedComponentError.__unicode__()`
 Inherited from `exceptions.BaseException`

exceptiontools.ExtraNoteHeadError

abjad.tools.exceptiontools.ExtraNoteHeadError

class tools.**ExtraNoteHeadError**

More than one note head found for single-note head operation.

Special Methods
 ExtraNoteHeadError.**__delattr__**()

 x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

 ExtraNoteHeadError.**__getitem__**()

 x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

 ExtraNoteHeadError.**__getslice__**()

 x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

 ExtraNoteHeadError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

 ExtraNoteHeadError.**__setattr__**()

 x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

 ExtraNoteHeadError.**__setstate__**()

Inherited from exceptions.BaseException

 ExtraNoteHeadError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

 ExtraNoteHeadError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.ExtraPitchError

abjad.tools.exceptiontools.ExtraPitchError

class tools.**ExtraPitchError**

More than one pitch found for single-pitch operation.

Special Methods

ExtraPitchError.**__delattr__**()
 x.**__delattr__**('name') <==> del x.name
 Inherited from exceptions.BaseException

ExtraPitchError.**__getitem__**()
 x.**__getitem__**(y) <==> x[y]
 Inherited from exceptions.BaseException

ExtraPitchError.**__getslice__**()
 x.**__getslice__**(i, j) <==> x[i:j]
 Use of negative indices is not supported.
 Inherited from exceptions.BaseException

ExtraPitchError.**__repr__**() <==> repr(x)
 Inherited from exceptions.BaseException

ExtraPitchError.**__setattr__**()
 x.**__setattr__**('name', value) <==> x.name = value
 Inherited from exceptions.BaseException

ExtraPitchError.**__setstate__**()
 Inherited from exceptions.BaseException

ExtraPitchError.**__str__**() <==> str(x)
 Inherited from exceptions.BaseException

ExtraPitchError.**__unicode__**()
 Inherited from exceptions.BaseException

exceptiontools.ExtraSpannerError

abjad.tools.exceptiontools.ExtraSpannerError

class tools.**ExtraSpannerError**

More than one spanner found for single-spanner operation.

Special Methods

ExtraSpannerError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

ExtraSpannerError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

ExtraSpannerError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

ExtraSpannerError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

ExtraSpannerError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

ExtraSpannerError.__setstate__()

Inherited from exceptions.BaseException

ExtraSpannerError.__str__() <==> str(x)

Inherited from exceptions.BaseException

ExtraSpannerError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.GraceContainerError

abjad.tools.exceptiontools.GraceContainerError

class tools.**GraceContainerError**

General grace container error.

Special Methods

GraceContainerError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

GraceContainerError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

GraceContainerError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

GraceContainerError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

GraceContainerError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

GraceContainerError.**__setstate__**()

Inherited from exceptions.BaseException

GraceContainerError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

GraceContainerError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.ImpreciseTempoError

abjad.tools.exceptiontools.ImpreciseTempoError

class tools.**ImpreciseTempoError**

TempoMark is imprecise.

Special Methods

ImpreciseTempoError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

ImpreciseTempoError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

ImpreciseTempoError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

ImpreciseTempoError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

ImpreciseTempoError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

ImpreciseTempoError.**__setstate__**()

Inherited from exceptions.BaseException

ImpreciseTempoError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

ImpreciseTempoError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.InputSpecificationError

abjad.tools.exceptiontools.InputSpecificationError

class tools.InputSpecificationError

Input is badly formed.

Special Methods

InputSpecificationError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

InputSpecificationError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

InputSpecificationError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

InputSpecificationError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

InputSpecificationError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

InputSpecificationError.__setstate__()

Inherited from exceptions.BaseException

InputSpecificationError.__str__() <==> str(x)

Inherited from exceptions.BaseException

InputSpecificationError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.InstrumentError

abjad.tools.exceptiontools.InstrumentError

class tools.InstrumentError

General instrument error.

Special Methods

```
InstrumentError.__delattr__ ()
    x.__delattr__('name') <==> del x.name
    Inherited from exceptions.BaseException
InstrumentError.__getitem__ ()
    x.__getitem__(y) <==> x[y]
    Inherited from exceptions.BaseException
InstrumentError.__getslice__ ()
    x.__getslice__(i, j) <==> x[i:j]
    Use of negative indices is not supported.
    Inherited from exceptions.BaseException
InstrumentError.__repr__ () <==> repr(x)
    Inherited from exceptions.BaseException
InstrumentError.__setattr__ ()
    x.__setattr__('name', value) <==> x.name = value
    Inherited from exceptions.BaseException
InstrumentError.__setstate__ ()
    Inherited from exceptions.BaseException
InstrumentError.__str__ () <==> str(x)
    Inherited from exceptions.BaseException
InstrumentError.__unicode__ ()
    Inherited from exceptions.BaseException
```

exceptiontools.IntervalError

abjad.tools.exceptiontools.IntervalError

class tools.IntervalError

General pitch interval error.

Special Methods

IntervalError.__delattr__()
 x.__delattr__('name') <==> del x.name
 Inherited from exceptions.BaseException

IntervalError.__getitem__()
 x.__getitem__(y) <==> x[y]
 Inherited from exceptions.BaseException

IntervalError.__getslice__()
 x.__getslice__(i, j) <==> x[i:j]
 Use of negative indices is not supported.
 Inherited from exceptions.BaseException

IntervalError.__repr__() <==> repr(x)
 Inherited from exceptions.BaseException

IntervalError.__setattr__()
 x.__setattr__('name', value) <==> x.name = value
 Inherited from exceptions.BaseException

IntervalError.__setstate__()
 Inherited from exceptions.BaseException

IntervalError.__str__() <==> str(x)
 Inherited from exceptions.BaseException

IntervalError.__unicode__()
 Inherited from exceptions.BaseException

exceptiontools.LilyPondParserError

abjad.tools.exceptiontools.LilyPondParserError

class tools.**LilyPondParserError**

Can not parse input.

Special Methods
 LilyPondParserError.**__delattr__**()

 x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

 LilyPondParserError.**__getitem__**()

 x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

 LilyPondParserError.**__getslice__**()

 x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

 LilyPondParserError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

 LilyPondParserError.**__setattr__**()

 x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

 LilyPondParserError.**__setstate__**()

Inherited from exceptions.BaseException

 LilyPondParserError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

 LilyPondParserError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.LineBreakError

abjad.tools.exceptiontools.LineBreakError

class tools.**LineBreakError**

General link break error.

Special Methods

LineBreakError.__delattr__()
 x.__delattr__('name') <==> del x.name
 Inherited from exceptions.BaseException

LineBreakError.__getitem__()
 x.__getitem__(y) <==> x[y]
 Inherited from exceptions.BaseException

LineBreakError.__getslice__()
 x.__getslice__(i, j) <==> x[i:j]
 Use of negative indices is not supported.
 Inherited from exceptions.BaseException

LineBreakError.__repr__() <==> repr(x)
 Inherited from exceptions.BaseException

LineBreakError.__setattr__()
 x.__setattr__('name', value) <==> x.name = value
 Inherited from exceptions.BaseException

LineBreakError.__setstate__()
 Inherited from exceptions.BaseException

LineBreakError.__str__() <==> str(x)
 Inherited from exceptions.BaseException

LineBreakError.__unicode__()
 Inherited from exceptions.BaseException

exceptiontools.MarkError

abjad.tools.exceptiontools.MarkError

class tools.**MarkError**

General mark error.

Special Methods

MarkError.**__delattr__**()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

MarkError.**__getitem__**()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

MarkError.**__getslice__**()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

MarkError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

MarkError.**__setattr__**()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

MarkError.**__setstate__**()

Inherited from exceptions.BaseException

MarkError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

MarkError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.MeasureContiguityError

abjad.tools.exceptiontools.MeasureContiguityError

class tools.MeasureContiguityError

Measures must be contiguous.

Special Methods

MeasureContiguityError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

MeasureContiguityError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

MeasureContiguityError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

MeasureContiguityError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

MeasureContiguityError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

MeasureContiguityError.__setstate__()

Inherited from exceptions.BaseException

MeasureContiguityError.__str__() <==> str(x)

Inherited from exceptions.BaseException

MeasureContiguityError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.MeasureError

abjad.tools.exceptiontools.MeasureError

class tools.**MeasureError**

General measure error.

Special Methods**MeasureError.__delattr__()**`x.__delattr__('name') <==> del x.name`Inherited from `exceptions.BaseException`**MeasureError.__getitem__()**`x.__getitem__(y) <==> x[y]`Inherited from `exceptions.BaseException`**MeasureError.__getslice__()**`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `exceptions.BaseException`**MeasureError.__repr__()** <==> `repr(x)`Inherited from `exceptions.BaseException`**MeasureError.__setattr__()**`x.__setattr__('name', value) <==> x.name = value`Inherited from `exceptions.BaseException`**MeasureError.__setstate__()**Inherited from `exceptions.BaseException`**MeasureError.__str__()** <==> `str(x)`Inherited from `exceptions.BaseException`**MeasureError.__unicode__()**Inherited from `exceptions.BaseException`

exceptiontools.MissingComponentError

abjad.tools.exceptiontools.MissingComponentError

class tools.**MissingComponentError**

No component found.

Special Methods

MissingComponentError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

MissingComponentError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

MissingComponentError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

MissingComponentError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

MissingComponentError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

MissingComponentError.**__setstate__**()

Inherited from exceptions.BaseException

MissingComponentError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

MissingComponentError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.MissingInstrumentError

abjad.tools.exceptiontools.MissingInstrumentError

```
class tools.MissingInstrumentError
```

```
    No instrument found.
```

Special Methods

```
MissingInstrumentError.__delattr__()
```

```
    x.__delattr__('name') <==> del x.name
```

```
    Inherited from exceptions.BaseException
```

```
MissingInstrumentError.__getitem__()
```

```
    x.__getitem__(y) <==> x[y]
```

```
    Inherited from exceptions.BaseException
```

```
MissingInstrumentError.__getslice__()
```

```
    x.__getslice__(i, j) <==> x[i:j]
```

```
    Use of negative indices is not supported.
```

```
    Inherited from exceptions.BaseException
```

```
MissingInstrumentError.__repr__() <==> repr(x)
```

```
    Inherited from exceptions.BaseException
```

```
MissingInstrumentError.__setattr__()
```

```
    x.__setattr__('name', value) <==> x.name = value
```

```
    Inherited from exceptions.BaseException
```

```
MissingInstrumentError.__setstate__()
```

```
    Inherited from exceptions.BaseException
```

```
MissingInstrumentError.__str__() <==> str(x)
```

```
    Inherited from exceptions.BaseException
```

```
MissingInstrumentError.__unicode__()
```

```
    Inherited from exceptions.BaseException
```

exceptiontools.MissingMarkError

`abjad.tools.exceptiontools.MissingMarkError`

class `tools.MissingMarkError`

No mark found.

Special Methods

`MissingMarkError.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Inherited from `exceptions.BaseException`

`MissingMarkError.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Inherited from `exceptions.BaseException`

`MissingMarkError.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `exceptions.BaseException`

`MissingMarkError.__repr__()` <==> `repr(x)`
Inherited from `exceptions.BaseException`

`MissingMarkError.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Inherited from `exceptions.BaseException`

`MissingMarkError.__setstate__()`
Inherited from `exceptions.BaseException`

`MissingMarkError.__str__()` <==> `str(x)`
Inherited from `exceptions.BaseException`

`MissingMarkError.__unicode__()`
Inherited from `exceptions.BaseException`

exceptiontools.MissingMeasureError`abjad.tools.exceptiontools.MissingMeasureError`**class** `tools.MissingMeasureError`

No measure found.

Special Methods`MissingMeasureError.__delattr__()``x.__delattr__('name') <==> del x.name`Inherited from `exceptions.BaseException``MissingMeasureError.__getitem__()``x.__getitem__(y) <==> x[y]`Inherited from `exceptions.BaseException``MissingMeasureError.__getslice__()``x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `exceptions.BaseException``MissingMeasureError.__repr__() <==> repr(x)`Inherited from `exceptions.BaseException``MissingMeasureError.__setattr__()``x.__setattr__('name', value) <==> x.name = value`Inherited from `exceptions.BaseException``MissingMeasureError.__setstate__()`Inherited from `exceptions.BaseException``MissingMeasureError.__str__() <==> str(x)`Inherited from `exceptions.BaseException``MissingMeasureError.__unicode__()`Inherited from `exceptions.BaseException`

exceptiontools.MissingNamedComponentError

abjad.tools.exceptiontools.MissingNamedComponentError

class tools.**MissingNamedComponentError**

No named component found.

Special Methods

MissingNamedComponentError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

MissingNamedComponentError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

MissingNamedComponentError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

MissingNamedComponentError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

MissingNamedComponentError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

MissingNamedComponentError.**__setstate__**()

Inherited from exceptions.BaseException

MissingNamedComponentError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

MissingNamedComponentError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.MissingNoteHeadError

abjad.tools.exceptiontools.MissingNoteHeadError

```
class tools.MissingNoteHeadError
```

No note head found.

Special Methods

```
MissingNoteHeadError.__delattr__()
```

`x.__delattr__('name') <==> del x.name`

Inherited from `exceptions.BaseException`

```
MissingNoteHeadError.__getitem__()
```

`x.__getitem__(y) <==> x[y]`

Inherited from `exceptions.BaseException`

```
MissingNoteHeadError.__getslice__()
```

`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `exceptions.BaseException`

```
MissingNoteHeadError.__repr__() <==> repr(x)
```

Inherited from `exceptions.BaseException`

```
MissingNoteHeadError.__setattr__()
```

`x.__setattr__('name', value) <==> x.name = value`

Inherited from `exceptions.BaseException`

```
MissingNoteHeadError.__setstate__()
```

Inherited from `exceptions.BaseException`

```
MissingNoteHeadError.__str__() <==> str(x)
```

Inherited from `exceptions.BaseException`

```
MissingNoteHeadError.__unicode__()
```

Inherited from `exceptions.BaseException`

exceptiontools.MissingPitchError

abjad.tools.exceptiontools.MissingPitchError

class tools.**MissingPitchError**

No pitch found.

Special Methods

MissingPitchError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

MissingPitchError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

MissingPitchError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

MissingPitchError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

MissingPitchError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

MissingPitchError.**__setstate__**()

Inherited from exceptions.BaseException

MissingPitchError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

MissingPitchError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.MissingSpannerError

abjad.tools.exceptiontools.MissingSpannerError

class tools.**MissingSpannerError**

No spanner found.

Special Methods

MissingSpannerError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

MissingSpannerError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

MissingSpannerError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

MissingSpannerError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

MissingSpannerError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

MissingSpannerError.__setstate__()

Inherited from exceptions.BaseException

MissingSpannerError.__str__() <==> str(x)

Inherited from exceptions.BaseException

MissingSpannerError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.MissingTempoError

`abjad.tools.exceptiontools.MissingTempoError`

class `tools.MissingTempoError`

No tempo found.

Special Methods

`MissingTempoError.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Inherited from `exceptions.BaseException`

`MissingTempoError.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `exceptions.BaseException`

`MissingTempoError.__getslice__()`

`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `exceptions.BaseException`

`MissingTempoError.__repr__() <==> repr(x)`

Inherited from `exceptions.BaseException`

`MissingTempoError.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Inherited from `exceptions.BaseException`

`MissingTempoError.__setstate__()`

Inherited from `exceptions.BaseException`

`MissingTempoError.__str__() <==> str(x)`

Inherited from `exceptions.BaseException`

`MissingTempoError.__unicode__()`

Inherited from `exceptions.BaseException`

exceptiontools.MusicContentsError

abjad.tools.exceptiontools.MusicContentsError

class tools.**MusicContentsError**

General container contents error.

Special Methods

MusicContentsError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

MusicContentsError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

MusicContentsError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

MusicContentsError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

MusicContentsError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

MusicContentsError.__setstate__()

Inherited from exceptions.BaseException

MusicContentsError.__str__() <==> str(x)

Inherited from exceptions.BaseException

MusicContentsError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.NegativeDurationError

`abjad.tools.exceptiontools.NegativeDurationError`

class `tools.NegativeDurationError`

Component duration must be positive.

Special Methods

`NegativeDurationError.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Inherited from `exceptions.BaseException`

`NegativeDurationError.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Inherited from `exceptions.BaseException`

`NegativeDurationError.__getslice__()`

`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `exceptions.BaseException`

`NegativeDurationError.__repr__() <==> repr(x)`

Inherited from `exceptions.BaseException`

`NegativeDurationError.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Inherited from `exceptions.BaseException`

`NegativeDurationError.__setstate__()`

Inherited from `exceptions.BaseException`

`NegativeDurationError.__str__() <==> str(x)`

Inherited from `exceptions.BaseException`

`NegativeDurationError.__unicode__()`

Inherited from `exceptions.BaseException`

exceptiontools.NonbinaryTimeSignatureConversionError

abjad.tools.exceptiontools.NonbinaryTimeSignatureConversionError

class tools.**NonbinaryTimeSignatureConversionError**

Nonbinary time signature has no binary equivalent.

Special Methods
 NonbinaryTimeSignatureConversionError.**__delattr__**()

 x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

 NonbinaryTimeSignatureConversionError.**__getitem__**()

 x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

 NonbinaryTimeSignatureConversionError.**__getslice__**()

 x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

 NonbinaryTimeSignatureConversionError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

 NonbinaryTimeSignatureConversionError.**__setattr__**()

 x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

 NonbinaryTimeSignatureConversionError.**__setstate__**()

Inherited from exceptions.BaseException

 NonbinaryTimeSignatureConversionError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

 NonbinaryTimeSignatureConversionError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.NonbinaryTimeSignatureSuppressionError

abjad.tools.exceptiontools.NonbinaryTimeSignatureSuppressionError

class tools.**NonbinaryTimeSignatureSuppressionError**

Suppressing nonbinary time signature will miscalculate duration of measure contents.

Special Methods

NonbinaryTimeSignatureSuppressionError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

NonbinaryTimeSignatureSuppressionError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

NonbinaryTimeSignatureSuppressionError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

NonbinaryTimeSignatureSuppressionError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

NonbinaryTimeSignatureSuppressionError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

NonbinaryTimeSignatureSuppressionError.**__setstate__**()

Inherited from exceptions.BaseException

NonbinaryTimeSignatureSuppressionError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

NonbinaryTimeSignatureSuppressionError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.NoteHeadError

abjad.tools.exceptiontools.NoteHeadError

class tools.**NoteHeadError**

General note head error.

Special MethodsNoteHeadError.**__delattr__**()x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

NoteHeadError.**__getitem__**()x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

NoteHeadError.**__getslice__**()x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

NoteHeadError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

NoteHeadError.**__setattr__**()x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

NoteHeadError.**__setstate__**()

Inherited from exceptions.BaseException

NoteHeadError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

NoteHeadError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.OverfullContainerError

abjad.tools.exceptiontools.OverfullContainerError

class tools.**OverfullContainerError**

Container contents duration is greater than container target duration.

Special Methods

OverfullContainerError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

OverfullContainerError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

OverfullContainerError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

OverfullContainerError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

OverfullContainerError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

OverfullContainerError.**__setstate__**()

Inherited from exceptions.BaseException

OverfullContainerError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

OverfullContainerError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.ParallelError

abjad.tools.exceptiontools.ParallelError

class tools.**ParallelError**

Parallel containers must contain contexts only.

Special Methods

ParallelError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

ParallelError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

ParallelError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

ParallelError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

ParallelError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

ParallelError.__setstate__()

Inherited from exceptions.BaseException

ParallelError.__str__() <==> str(x)

Inherited from exceptions.BaseException

ParallelError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.PartitionError

abjad.tools.exceptiontools.PartitionError

class tools.**PartitionError**

General partition error.

Special Methods

PartitionError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

PartitionError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

PartitionError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

PartitionError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

PartitionError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

PartitionError.__setstate__()

Inherited from exceptions.BaseException

PartitionError.__str__() <==> str(x)

Inherited from exceptions.BaseException

PartitionError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.PitchError

abjad.tools.exceptiontools.PitchError

class tools.**PitchError**

General pitch error.

Special Methods**PitchError.__delattr__()**`x.__delattr__('name') <==> del x.name`Inherited from `exceptions.BaseException`**PitchError.__getitem__()**`x.__getitem__(y) <==> x[y]`Inherited from `exceptions.BaseException`**PitchError.__getslice__()**`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Inherited from `exceptions.BaseException`**PitchError.__repr__()** <==> `repr(x)`Inherited from `exceptions.BaseException`**PitchError.__setattr__()**`x.__setattr__('name', value) <==> x.name = value`Inherited from `exceptions.BaseException`**PitchError.__setstate__()**Inherited from `exceptions.BaseException`**PitchError.__str__()** <==> `str(x)`Inherited from `exceptions.BaseException`**PitchError.__unicode__()**Inherited from `exceptions.BaseException`

exceptiontools.SchemeParserFinishedException

abjad.tools.exceptiontools.SchemeParserFinishedException

class tools.**SchemeParserFinishedException**

SchemeParser has finished parsing.

Special Methods

SchemeParserFinishedException.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

SchemeParserFinishedException.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

SchemeParserFinishedException.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

SchemeParserFinishedException.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

SchemeParserFinishedException.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

SchemeParserFinishedException.**__setstate__**()

Inherited from exceptions.BaseException

SchemeParserFinishedException.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

SchemeParserFinishedException.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.SpacingError

abjad.tools.exceptiontools.SpacingError

class tools.**SpacingError**

General spacing error.

Special Methods

SpacingError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

SpacingError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

SpacingError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

SpacingError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

SpacingError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

SpacingError.__setstate__()

Inherited from exceptions.BaseException

SpacingError.__str__() <==> str(x)

Inherited from exceptions.BaseException

SpacingError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.SpannerError

abjad.tools.exceptiontools.SpannerError

class tools.**SpannerError**

General spanner error.

Special Methods

SpannerError.__delattr__()
 x.__delattr__('name') <==> del x.name
 Inherited from exceptions.BaseException

SpannerError.__getitem__()
 x.__getitem__(y) <==> x[y]
 Inherited from exceptions.BaseException

SpannerError.__getslice__()
 x.__getslice__(i, j) <==> x[i:j]
 Use of negative indices is not supported.
 Inherited from exceptions.BaseException

SpannerError.__repr__() <==> repr(x)
 Inherited from exceptions.BaseException

SpannerError.__setattr__()
 x.__setattr__('name', value) <==> x.name = value
 Inherited from exceptions.BaseException

SpannerError.__setstate__()
 Inherited from exceptions.BaseException

SpannerError.__str__() <==> str(x)
 Inherited from exceptions.BaseException

SpannerError.__unicode__()
 Inherited from exceptions.BaseException

exceptiontools.SpannerPopulationError

abjad.tools.exceptiontools.SpannerPopulationError

class tools.**SpannerPopulationError**

Spanner contents incorrect.

Spanner may be missing component it is assumed to have.

Spanner may have a component it is assumed not to have.

Special Methods

SpannerPopulationError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

SpannerPopulationError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

SpannerPopulationError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

SpannerPopulationError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

SpannerPopulationError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

SpannerPopulationError.**__setstate__**()

Inherited from exceptions.BaseException

SpannerPopulationError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

SpannerPopulationError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.StaffContainmentError

abjad.tools.exceptiontools.StaffContainmentError

class tools.StaffContainmentError
 Staves must not contain staff groups, scores or other staves.

Special Methods

StaffContainmentError.__delattr__()
 x.__delattr__('name') <==> del x.name
 Inherited from exceptions.BaseException

StaffContainmentError.__getitem__()
 x.__getitem__(y) <==> x[y]
 Inherited from exceptions.BaseException

StaffContainmentError.__getslice__()
 x.__getslice__(i, j) <==> x[i:j]
 Use of negative indices is not supported.
 Inherited from exceptions.BaseException

StaffContainmentError.__repr__() <==> repr(x)
 Inherited from exceptions.BaseException

StaffContainmentError.__setattr__()
 x.__setattr__('name', value) <==> x.name = value
 Inherited from exceptions.BaseException

StaffContainmentError.__setstate__()
 Inherited from exceptions.BaseException

StaffContainmentError.__str__() <==> str(x)
 Inherited from exceptions.BaseException

StaffContainmentError.__unicode__()
 Inherited from exceptions.BaseException

exceptiontools.TempoError

abjad.tools.exceptiontools.TempoError

class tools.**TempoError**

General tempo error.

Special Methods**TempoError.__delattr__()****x.__delattr__('name')** <==> del x.nameInherited from `exceptions.BaseException`**TempoError.__getitem__()****x.__getitem__(y)** <==> x[y]Inherited from `exceptions.BaseException`**TempoError.__getslice__()****x.__getslice__(i, j)** <==> x[i:j]

Use of negative indices is not supported.

Inherited from `exceptions.BaseException`**TempoError.__repr__()** <==> *repr*(x)Inherited from `exceptions.BaseException`**TempoError.__setattr__()****x.__setattr__('name', value)** <==> x.name = valueInherited from `exceptions.BaseException`**TempoError.__setstate__()**Inherited from `exceptions.BaseException`**TempoError.__str__()** <==> *str*(x)Inherited from `exceptions.BaseException`**TempoError.__unicode__()**Inherited from `exceptions.BaseException`

exceptiontools.TieChainError

abjad.tools.exceptiontools.TieChainError

class tools.**TieChainError**

General tie chain error.

Special Methods

TieChainError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

TieChainError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

TieChainError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

TieChainError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

TieChainError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

TieChainError.**__setstate__**()

Inherited from exceptions.BaseException

TieChainError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

TieChainError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.TimeSignatureAssignmentError

abjad.tools.exceptiontools.TimeSignatureAssignmentError

class tools.**TimeSignatureAssignmentError**

Can not assign time signature to dynamic measure.

Special Methods

TimeSignatureAssignmentError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

TimeSignatureAssignmentError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

TimeSignatureAssignmentError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

TimeSignatureAssignmentError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

TimeSignatureAssignmentError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

TimeSignatureAssignmentError.**__setstate__**()

Inherited from exceptions.BaseException

TimeSignatureAssignmentError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

TimeSignatureAssignmentError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.TimeSignatureError

abjad.tools.exceptiontools.TimeSignatureError

class tools.**TimeSignatureError**

General time signature error.

Special Methods

TimeSignatureError.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

TimeSignatureError.**__getitem__**()

x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

TimeSignatureError.**__getslice__**()

x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

TimeSignatureError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

TimeSignatureError.**__setattr__**()

x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

TimeSignatureError.**__setstate__**()

Inherited from exceptions.BaseException

TimeSignatureError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

TimeSignatureError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.TonalHarmonyError

abjad.tools.exceptiontools.TonalHarmonyError

class tools.**TonalHarmonyError**

General tonal harmony error.

Special MethodsTonalHarmonyError.**__delattr__**()x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

TonalHarmonyError.**__getitem__**()x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

TonalHarmonyError.**__getslice__**()x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

TonalHarmonyError.**__repr__**() <==> repr(x)

Inherited from exceptions.BaseException

TonalHarmonyError.**__setattr__**()x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

TonalHarmonyError.**__setstate__**()

Inherited from exceptions.BaseException

TonalHarmonyError.**__str__**() <==> str(x)

Inherited from exceptions.BaseException

TonalHarmonyError.**__unicode__**()

Inherited from exceptions.BaseException

exceptiontools.TupletError

abjad.tools.exceptiontools.TupletError

class tools.TupletError

Geneal tuplet error.

Special Methods

TupletError.__delattr__()
 x.__delattr__('name') <==> del x.name
 Inherited from exceptions.BaseException

TupletError.__getitem__()
 x.__getitem__(y) <==> x[y]
 Inherited from exceptions.BaseException

TupletError.__getslice__()
 x.__getslice__(i, j) <==> x[i:j]
 Use of negative indices is not supported.
 Inherited from exceptions.BaseException

TupletError.__repr__() <==> repr(x)
 Inherited from exceptions.BaseException

TupletError.__setattr__()
 x.__setattr__('name', value) <==> x.name = value
 Inherited from exceptions.BaseException

TupletError.__setstate__()
 Inherited from exceptions.BaseException

TupletError.__str__() <==> str(x)
 Inherited from exceptions.BaseException

TupletError.__unicode__()
 Inherited from exceptions.BaseException

exceptiontools.TupletFuseError

abjad.tools.exceptiontools.TupletFuseError

class tools.TupletFuseError

Tuplets must carry same multiplier and be same type in order to fuse correctly.

Special Methods

`TupletFuseError.__delattr__()`
`x.__delattr__('name') <==> del x.name`
 Inherited from `exceptions.BaseException`

`TupletFuseError.__getitem__()`
`x.__getitem__(y) <==> x[y]`
 Inherited from `exceptions.BaseException`

`TupletFuseError.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
 Use of negative indices is not supported.
 Inherited from `exceptions.BaseException`

`TupletFuseError.__repr__()` <==> `repr(x)`
 Inherited from `exceptions.BaseException`

`TupletFuseError.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`
 Inherited from `exceptions.BaseException`

`TupletFuseError.__setstate__()`
 Inherited from `exceptions.BaseException`

`TupletFuseError.__str__()` <==> `str(x)`
 Inherited from `exceptions.BaseException`

`TupletFuseError.__unicode__()`
 Inherited from `exceptions.BaseException`

exceptiontools.TypographicWhitespaceError

abjad.tools.exceptiontools.TypographicWhitespaceError

class tools.**TypographicWhitespaceError**
 Whitespace after leaf confuses LilyPond timekeeping.

Special Methods

TypographicWhitespaceError.**__delattr__**()
 x.**__delattr__**('name') <==> del x.name

Inherited from exceptions.BaseException

TypographicWhitespaceError.**__getitem__**()
 x.**__getitem__**(y) <==> x[y]

Inherited from exceptions.BaseException

TypographicWhitespaceError.**__getslice__**()
 x.**__getslice__**(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

TypographicWhitespaceError.**__repr__**() <==> repr(x)
 Inherited from exceptions.BaseException

TypographicWhitespaceError.**__setattr__**()
 x.**__setattr__**('name', value) <==> x.name = value

Inherited from exceptions.BaseException

TypographicWhitespaceError.**__setstate__**()
 Inherited from exceptions.BaseException

TypographicWhitespaceError.**__str__**() <==> str(x)
 Inherited from exceptions.BaseException

TypographicWhitespaceError.**__unicode__**()
 Inherited from exceptions.BaseException

exceptiontools.UnboundedTimeIntervalError

abjad.tools.exceptiontools.UnboundedTimeIntervalError

class tools.UnboundedTimeIntervalError

Time interval has no bounds.

Special Methods

UnboundedTimeIntervalError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

UnboundedTimeIntervalError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

UnboundedTimeIntervalError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

UnboundedTimeIntervalError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

UnboundedTimeIntervalError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

UnboundedTimeIntervalError.__setstate__()

Inherited from exceptions.BaseException

UnboundedTimeIntervalError.__str__() <==> str(x)

Inherited from exceptions.BaseException

UnboundedTimeIntervalError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.UndefinedSpacingError

abjad.tools.exceptiontools.UndefinedSpacingError

class tools.UndefinedSpacingError

No spacing value found.

Special Methods

UndefinedSpacingError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

UndefinedSpacingError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

UndefinedSpacingError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

UndefinedSpacingError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

UndefinedSpacingError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

UndefinedSpacingError.__setstate__()

Inherited from exceptions.BaseException

UndefinedSpacingError.__str__() <==> str(x)

Inherited from exceptions.BaseException

UndefinedSpacingError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.UnderfullContainerError

abjad.tools.exceptiontools.UnderfullContainerError

class tools.UnderfullContainerError

Container contents duration is less than container target duration.

Special Methods

UnderfullContainerError.__delattr__()

x.__delattr__('name') <==> del x.name

Inherited from exceptions.BaseException

UnderfullContainerError.__getitem__()

x.__getitem__(y) <==> x[y]

Inherited from exceptions.BaseException

UnderfullContainerError.__getslice__()

x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Inherited from exceptions.BaseException

UnderfullContainerError.__repr__() <==> repr(x)

Inherited from exceptions.BaseException

UnderfullContainerError.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Inherited from exceptions.BaseException

UnderfullContainerError.__setstate__()

Inherited from exceptions.BaseException

UnderfullContainerError.__str__() <==> str(x)

Inherited from exceptions.BaseException

UnderfullContainerError.__unicode__()

Inherited from exceptions.BaseException

exceptiontools.VoiceContainmentError

abjad.tools.exceptiontools.VoiceContainmentError

class tools.VoiceContainmentError
 Voice must not contain staves, staff groups or scores.

Special Methods

VoiceContainmentError.__delattr__()
 x.__delattr__('name') <==> del x.name
 Inherited from exceptions.BaseException

VoiceContainmentError.__getitem__()
 x.__getitem__(y) <==> x[y]
 Inherited from exceptions.BaseException

VoiceContainmentError.__getslice__()
 x.__getslice__(i, j) <==> x[i:j]
 Use of negative indices is not supported.
 Inherited from exceptions.BaseException

VoiceContainmentError.__repr__() <==> repr(x)
 Inherited from exceptions.BaseException

VoiceContainmentError.__setattr__()
 x.__setattr__('name', value) <==> x.name = value
 Inherited from exceptions.BaseException

VoiceContainmentError.__setstate__()
 Inherited from exceptions.BaseException

VoiceContainmentError.__str__() <==> str(x)
 Inherited from exceptions.BaseException

VoiceContainmentError.__unicode__()
 Inherited from exceptions.BaseException

formattools

functions

formattools.get_all_format_contributions

formattools.get_all_format_contributions(*component*)
 Get all format contributions for *component* as a nested dictionary structure.

Return dict.

formattools.get_all_mark_format_contributions

`formattools.get_all_mark_format_contributions` (*component*)

Get all mark format contributions as nested dictionaries.

The first level of keys represent format slots.

The second level of keys represent format contributor ('articulations', 'markup', etc.).

Return dict.

formattools.get_articulation_format_contributions

`formattools.get_articulation_format_contributions` (*component*)

New in version 2.0. Get articulation format contributions for *component*.

Return list.

formattools.get_comment_format_contributions_for_slot

`formattools.get_comment_format_contributions_for_slot` (*component*, *slot*)

New in version 2.0. Get comment format contributions for *component* at *slot*.

Return list.

formattools.get_context_mark_format_contributions_for_slot

`formattools.get_context_mark_format_contributions_for_slot` (*component*, *slot*)

New in version 2.0. Get context mark format contributions for *component* at *slot*.

Return list.

formattools.get_context_mark_format_pieces

`formattools.get_context_mark_format_pieces` (*mark*)

Get context mark format pieces for *mark*.

Return list.

formattools.get_context_setting_format_contributions

`formattools.get_context_setting_format_contributions` (*component*)

New in version 2.0. Get context setting format contributions for *component*.

Return sorted list.

formattools.get_grob_override_format_contributions

`formattools.get_grob_override_format_contributions` (*component*)

New in version 2.0. Get grob override format contributions for *component*.

Return alphabetized list of LilyPond grob overrides.

formattools.get_grob_revert_format_contributions

`formattools.get_grob_revert_format_contributions` (*component*)

New in version 2.0. Get grob revert format contributions.

Return alphabetized list of LilyPond grob reverts.

formattools.get_lilypond_command_mark_format_contributions_for_slot

`formattools.get_lilypond_command_mark_format_contributions_for_slot` (*component*,
slot)

New in version 2.0. Get LilyPond command mark format contributions for *component* at *slot*.

Return list.

formattools.get_markup_format_contributions

`formattools.get_markup_format_contributions` (*component*)

New in version 2.0. Get markup format contributions for *component*.

Return list.

formattools.get_spanner_format_contributions

`formattools.get_spanner_format_contributions` (*leaf*)

Get spanner format contributions for *leaf* as a dictionary of format_slot:contributions key:value pairs.

Return dict.

formattools.get_spanner_format_contributions_for_slot

`formattools.get_spanner_format_contributions_for_slot` (*leaf*, *slot*)

New in version 2.0. Get spanner format contributions for *leaf* at *slot*.

Return list.

formattools.get_stem_tremolo_format_contributions

`formattools.get_stem_tremolo_format_contributions` (*component*)

New in version 2.0. Get stem tremolo format contributions for *component*.

Return list.

formattools.is_formattable_context_mark_for_component

`formattools.is_formattable_context_mark_for_component` (*mark*, *component*)

Return True if ContextMark *mark* can format for *component*.

formattools.report_spanner_format_contributions

`formattools.report_spanner_format_contributions` (*spanner*, *screen=True*)

New in version 2.9. Report spanner format contributions for every leaf to which spanner attaches.

```
>>> staff = Staff("c8 d e f")
>>> spanner = beamtools.BeamSpanner(staff[:])

>>> formattools.report_spanner_format_contributions(spanner)
c8 before: []
   after: []
   right: ['[']

d8 before: []
   after: []
   right: []

e8 before: []
   after: []
   right: []

f8 before: []
   after: []
   right: [']']
```

Return none or return string.

importtools**functions****importtools.import_structured_package**

`importtools.import_structured_package` (*path*, *namespace*, *package_root_name='abjad'*)

New in version 2.9. Import public names from *path* into *namespace*.

This is the custom function that all Abjad packages use to import public classes and functions on startup.

The function will work for any package laid out like Abjad packages.

Set *package_root_name* to the root any Abjad-like package structure.

Return none.

introspectiontools

functions

introspectiontools.get_current_function_name

`introspectiontools.get_current_function_name()`

New in version 2.10. Get current function name:

```
>>> from abjad.tools import introspectiontools

>>> def foo():
...     function_name = introspectiontools.get_current_function_name()
...     print 'Function name is {!r}'.format(function_name)

>>> foo()
Function name is 'foo'.
```

Call this function within the implementation of any ofther function.

Returns enclosing function name as a string or else none.

introspectiontools.klass_to_tools_package_qualified_class_name

`introspectiontools.klass_to_tools_package_qualified_class_name(klass)`

New in version 2.10. Change *klass* to tools package-qualified class name:

```
>>> from abjad.tools import introspectiontools

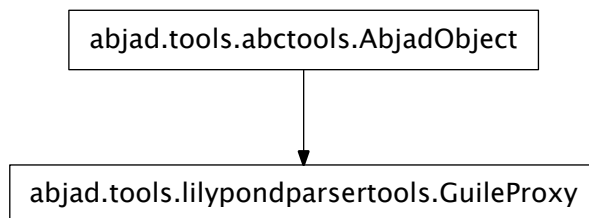
>>> introspectiontools.klass_to_tools_package_qualified_class_name(Note)
'notetools.Note'
```

Return string.

lilypondparsertools

concrete classes

lilypondparsertools.GuileProxy



class `lilypondparsertools.GuileProxy` (*client*)
Emulates LilyPond music functions.
Used internally by LilyPondParser.
Not composer-safe.

Read-only Properties

`GuileProxy.storage_format`
Storage format of Abjad object.
Return string.
Inherited from `abctools.AbjadObject`

Methods

`GuileProxy.acciaccatura` (*music*)
`GuileProxy.appoggiatura` (*music*)
`GuileProxy.bar` (*string*)
`GuileProxy.breathe` ()
`GuileProxy.clef` (*string*)
`GuileProxy.grace` (*music*)
`GuileProxy.key` (*notename_pitch, number_list*)
`GuileProxy.language` (*string*)
`GuileProxy.makeClusters` (*music*)
`GuileProxy.mark` (*label*)
`GuileProxy.oneVoice` ()
`GuileProxy.relative` (*pitch, music*)
`GuileProxy.skip` (*duration*)
`GuileProxy.slashedGraceContainer` (*music*)
`GuileProxy.time` (*number_list, fraction*)
`GuileProxy.times` (*fraction, music*)
`GuileProxy.transpose` (*from_pitch, to_pitch, music*)
`GuileProxy.voiceFour` ()
`GuileProxy.voiceOne` ()
`GuileProxy.voiceThree` ()
`GuileProxy.voiceTwo` ()

Special Methods

`GuileProxy.__call__` (*function_name, args*)
`GuileProxy.__eq__` (*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.
Inherited from `abctools.AbjadObject`

`GuileProxy.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GuileProxy.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`GuileProxy.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GuileProxy.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`GuileProxy.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

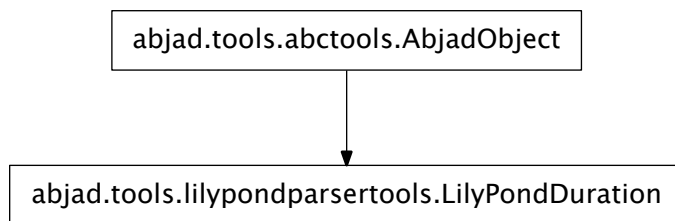
`GuileProxy.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`lilypondparsertools.LilyPondDuration`



class `lilypondparsertools.LilyPondDuration` (*duration, multiplier=None*)

Model of a duration in LilyPond.

Not composer-safe.

Used internally by LilyPondParser.

Read-only Properties

`LilyPondDuration.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`LilyPondDuration.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`LilyPondDuration.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondDuration.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`LilyPondDuration.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondDuration.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondDuration.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

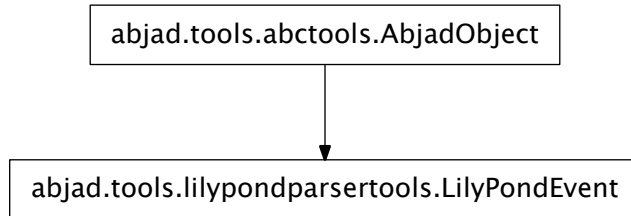
`LilyPondDuration.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`lilypondparsertools.LilyPondEvent`



class `lilypondparsertools.LilyPondEvent` (*name*, ***kwargs*)

Model of an arbitrary event in LilyPond.

Not composer-safe.

Used internally by LilyPondParser.

Read-only Properties

`LilyPondEvent.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`LilyPondEvent.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`LilyPondEvent.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondEvent.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`LilyPondEvent.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondEvent.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

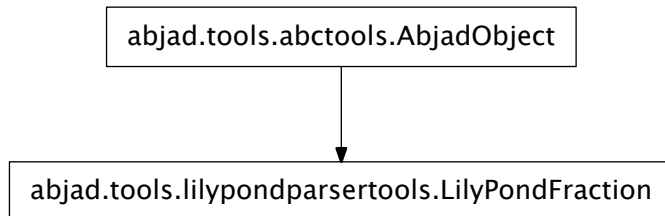
`LilyPondEvent.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`LilyPondEvent.__repr__()`

`lilypondparsertools.LilyPondFraction`



class `lilypondparsertools.LilyPondFraction` (*numerator, denominator*)
 Model of a fraction in LilyPond.

Not composer-safe.

Used internally by `LilyPondParser`.

Read-only Properties

`LilyPondFraction.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`LilyPondFraction.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`LilyPondFraction.__ge__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondFraction.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`LilyPondFraction.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondFraction.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondFraction.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

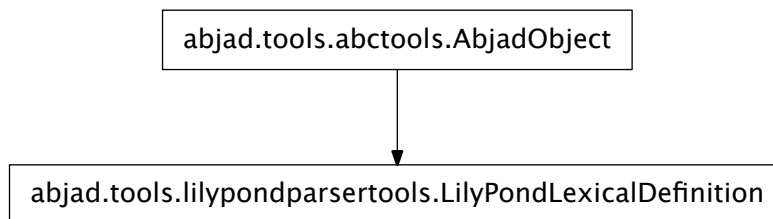
`LilyPondFraction.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`lilypondparsertools.LilyPondLexicalDefinition`



class `lilypondparsertools.LilyPondLexicalDefinition` (*client*)

The lexical definition of LilyPond's syntax.

Effectively equivalent to LilyPond's `lexer.ll` file.

Not composer-safe.

Used internally by `LilyPondParser`.

Read-only Properties

`LilyPondLexicalDefinition.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`LilyPondLexicalDefinition.push_signature(signature, t)`

`LilyPondLexicalDefinition.scan_bare_word(t)`

`LilyPondLexicalDefinition.scan_escaped_word(t)`

`LilyPondLexicalDefinition.t_651_a(t)`
 $(((-?[0-9]^+).[0-9]^+)(-?[0-9]^+))$

`LilyPondLexicalDefinition.t_651_b(t)`
 $-[0-9]^+$

`LilyPondLexicalDefinition.t_661(t)`
 $-.$

`LilyPondLexicalDefinition.t_666(t)`
 $[0-9]^+$

`LilyPondLexicalDefinition.t_ANY_165(t)`
 r

`LilyPondLexicalDefinition.t_INITIAL_643(t)`
 $[a-zA-Z200-377]((([a-zA-Z200-377]_)|[0-9])|-)^*$

`LilyPondLexicalDefinition.t_INITIAL_646(t)`
 $\backslash[a-zA-Z200-377]((([a-zA-Z200-377]_)|[0-9])|-)^*$

`LilyPondLexicalDefinition.t_INITIAL_markup_notes_210(t)`
 $\%{\$

`LilyPondLexicalDefinition.t_INITIAL_markup_notes_214(t)`
 $\%[\wedge\{nr\}[\wedge nr]^*\{nr\}$

`LilyPondLexicalDefinition.t_INITIAL_markup_notes_216(t)`
 $\%[\wedge\{nr\}$

`LilyPondLexicalDefinition.t_INITIAL_markup_notes_218(t)`
 $\%[nr]$

`LilyPondLexicalDefinition.t_INITIAL_markup_notes_220(t)`
 $\%[\wedge\{nr\}[\wedge nr]^*$

`LilyPondLexicalDefinition.t_INITIAL_markup_notes_222(t)`
 $[$
 $]$

`LilyPondLexicalDefinition.t_INITIAL_markup_notes_227(t)`
 $“$

`LilyPondLexicalDefinition.t_INITIAL_markup_notes_353(t)`
 $\#$

`LilyPondLexicalDefinition.t_INITIAL_markup_notes_353_boolean(t)`
 $\#\#(tlf)$

```

LilyPondLexicalDefinition.t_INITIAL_markup_notes_353_identifier(t)
    #'[a-zA-Z200-377]((([a-zA-Z200-377]|_)|[0-9])|~)*

LilyPondLexicalDefinition.t_INITIAL_notes_233(t)
    \version[ ntfr]*

LilyPondLexicalDefinition.t_INITIAL_notes_387(t)
    <<

LilyPondLexicalDefinition.t_INITIAL_notes_390(t)
    >>

LilyPondLexicalDefinition.t_INITIAL_notes_396(t)
    <

LilyPondLexicalDefinition.t_INITIAL_notes_399(t)
    >

LilyPondLexicalDefinition.t_INITIAL_notes_686(t)
    \.

LilyPondLexicalDefinition.t_error(t)

LilyPondLexicalDefinition.t_longcomment_291(t)
    [^%]+

LilyPondLexicalDefinition.t_longcomment_293(t)
    %+[^}%]*

LilyPondLexicalDefinition.t_longcomment_296(t)
    %}

LilyPondLexicalDefinition.t_longcomment_error(t)

LilyPondLexicalDefinition.t_markup_545(t)
    \score

LilyPondLexicalDefinition.t_markup_548(t)
    \([a-zA-Z200-377]|[-_])+

LilyPondLexicalDefinition.t_markup_601(t)
    [^#{ }'\tnrf]+

LilyPondLexicalDefinition.t_markup_error(t)

LilyPondLexicalDefinition.t_newline(t)
    n+

LilyPondLexicalDefinition.t_notes_417(t)
    [a-zA-Z200-377]+

LilyPondLexicalDefinition.t_notes_421(t)
    \[a-zA-Z200-377]+

LilyPondLexicalDefinition.t_notes_424(t)
    [0-9]+/[0-9]+

LilyPondLexicalDefinition.t_notes_428(t)
    [0-9]+//

LilyPondLexicalDefinition.t_notes_428b(t)
    [0-9]+

LilyPondLexicalDefinition.t_notes_433(t)
    \[0-9]+

```

```

LilyPondLexicalDefinition.t_notes_error(t)
LilyPondLexicalDefinition.t_quote_440(t)
    [nt\''']
LilyPondLexicalDefinition.t_quote_443(t)
    [^\'"]+
LilyPondLexicalDefinition.t_quote_446(t)
    “
LilyPondLexicalDefinition.t_quote_456(t)
    .
LilyPondLexicalDefinition.t_quote_XXX(t)
    \”
LilyPondLexicalDefinition.t_quote_error(t)
LilyPondLexicalDefinition.t_scheme_error(t)
LilyPondLexicalDefinition.t_version_242(t)
    “[^”]*”
LilyPondLexicalDefinition.t_version_278(t)
    (.ln)
LilyPondLexicalDefinition.t_version_341(t)
    “[^”]*
LilyPondLexicalDefinition.t_version_error(t)

```

Special Methods

```

LilyPondLexicalDefinition.__eq__(arg)
    True when id(self) equals id(arg).

    Return boolean.

    Inherited from abctools.AbjadObject
LilyPondLexicalDefinition.__ge__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject
LilyPondLexicalDefinition.__gt__(arg)
    Abjad objects by default do not implement this method.

    Raise exception

    Inherited from abctools.AbjadObject
LilyPondLexicalDefinition.__le__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

    Inherited from abctools.AbjadObject
LilyPondLexicalDefinition.__lt__(arg)
    Abjad objects by default do not implement this method.

    Raise exception.

```

Inherited from `abctools.AbjadObject`

`LilyPondLexicalDefinition.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

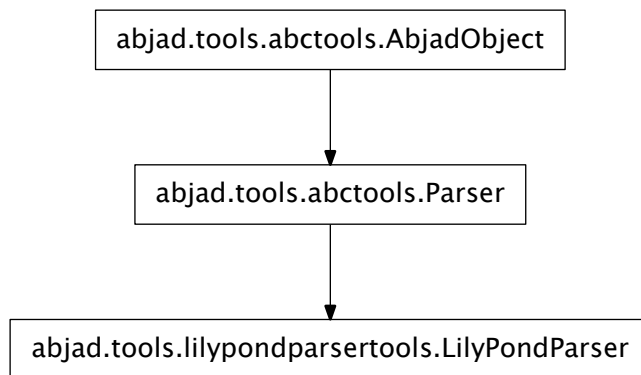
Return boolean.

Inherited from `abctools.AbjadObject`

`LilyPondLexicalDefinition.__repr__()`
 Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
 Return string.

Inherited from `abctools.AbjadObject`

`lilypondparsertools.LilyPondParser`



class `lilypondparsertools.LilyPondParser` (*default_language='english', debug=False*)
 Parses a subset of LilyPond input syntax:

```

>>> from abjad.tools.lilypondparsertools import LilyPondParser
>>> parser = LilyPondParser()
>>> input = r"\new Staff { c'4 ( d'8 e' fs'2) \fermata }"
>>> result = parser(input)
>>> f(result)
\new Staff {
  c'4 (
  d'8
  e'8
  fs'2 -\fermata )
}
  
```

`LilyPondParser` defaults to English note names, but any of the other languages supported by LilyPond may be used:

```

>>> parser = LilyPondParser('nederlands')
>>> input = '{ c des e fis }'
>>> result = parser(input)
  
```

```
>>> f(result)
{
    c4
    df4
    e4
    fs4
}
```

Briefly, LilyPondParser understands theses aspects of LilyPond syntax:

- Notes, chords, rests, skips and multi-measure rests
- Durations, dots, and multipliers
- All pitchnames, and octave ticks
- Simple markup (i.e. `c'4 ^ "hello!"`)
- Most articulations
- Most spanners, including beams, slurs, phrasing slurs, ties, and glissandi
- Most context types via `\new` and `\context`, as well as context ids (i.e. `\new Staff = "foo" { }`)
- Variable assignment (i.e. `global = { \time 3/4 } \new Staff { \global }`)
- Many music functions:
 - `\acciaccatura`
 - `\appoggiatura`
 - `\bar`
 - `\breathe`
 - `\clef`
 - `\grace`
 - `\key`
 - `\transpose`
 - `\language`
 - `\makeClusters`
 - `\mark`
 - `\oneVoice`
 - `\relative`
 - `\skip`
 - `\slashedGrace`
 - `\time`
 - `\times`
 - `\transpose`
 - `\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`

LilyPondParser currently **DOES NOT** understand many other aspects of LilyPond syntax:

- `\markup`
- `\book`, `\bookpart`, `\header`, `\layout`, `\midi` and `\paper`
- `\repeat` and `\alternative`
- Lyrics
- `\chordmode`, `\drummode` or `\figuremode`
- Property operations, such as `\override`, `\revert`, `\set`, `\unset`, and `\once`
- Music functions which generate or extensively mutate musical structures
- Embedded Scheme statements (anything beginning with `#`)

Returns LilyPondParser instance.

Read-only Properties

`LilyPondParser.available_languages`

Tuple of pitch-name languages supported by LilyPondParser:

```
>>> from abjad.tools.lilypondparsertools import LilyPondParser
>>> parser = LilyPondParser()
>>> parser.available_languages
('catalan', 'deutsch', 'english', 'espanol', 'italiano', 'nederlands',
'norsk', 'portugues', 'suomi', 'svenska', 'vlaams')
```

Return tuple.

`LilyPondParser.debug`

True if the parser runs in debugging mode.

Inherited from `abctools.Parser`

`LilyPondParser.lexer`

The parser's PLY Lexer instance.

Inherited from `abctools.Parser`

`LilyPondParser.lexer_rules_object`

`LilyPondParser.logger`

The parser's Logger instance.

Inherited from `abctools.Parser`

`LilyPondParser.logger_path`

The output path for the parser's logfile.

Inherited from `abctools.Parser`

`LilyPondParser.output_path`

The output path for files associated with the parser.

Inherited from `abctools.Parser`

`LilyPondParser.parser`

The parser's PLY LRParser instance.

Inherited from `abctools.Parser`

`LilyPondParser.parser_rules_object`

`LilyPondParser.pickle_path`

The output path for the parser's pickled parsing tables.

Inherited from `abctools.Parser`

`LilyPondParser.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Read/write Properties

`LilyPondParser.default_language`

Read/write attribute to set parser's default pitch-name language:

```
>>> from abjad.tools.lilypondparsertools import LilyPondParser
>>> parser = LilyPondParser()
>>> parser.default_language
'english'
>>> parser('{ c df e fs }')
{c4, df4, e4, fs4}
>>> parser.default_language = 'nederlands'
>>> parser.default_language
'nederlands'
>>> parser('{ c des e fis }')
{c4, df4, e4, fs4}
```

Methods

`LilyPondParser.tokenize(input_string)`

Tokenize *input string* and print results.

Inherited from `abctools.Parser`

Special Methods

`LilyPondParser.__call__(input_string)`

`LilyPondParser.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`LilyPondParser.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondParser.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`LilyPondParser.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondParser.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondParser.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

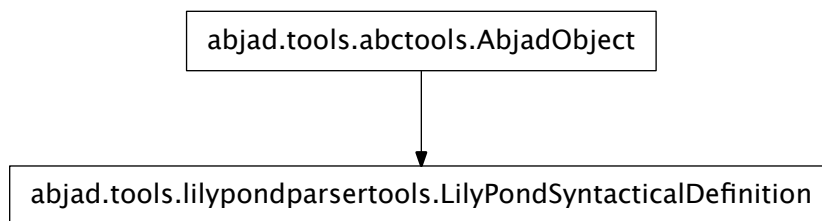
`LilyPondParser.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`lilypondparsertools.LilyPondSyntacticalDefinition`



class `lilypondparsertools.LilyPondSyntacticalDefinition` (*client*)

The syntactical definition of LilyPond's syntax.

Effectively equivalent to LilyPond's `parser.yy` file.

Not composer-safe.

Used internally by `LilyPondParser`.

Read-only Properties

`LilyPondSyntacticalDefinition.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

```

LilyPondSyntacticalDefinition.p_assignment__assignment_id__Chr61__identifier_init(p)
    assignment : assignment_id '=' identifier_init

LilyPondSyntacticalDefinition.p_assignment__embedded_scm(p)
    assignment : embedded_scm

LilyPondSyntacticalDefinition.p_assignment_id__STRING(p)
    assignment_id : STRING

LilyPondSyntacticalDefinition.p_bare_number__REAL__NUMBER_IDENTIFIER(p)
    bare_number : REAL NUMBER_IDENTIFIER

LilyPondSyntacticalDefinition.p_bare_number__UNSIGNED__NUMBER_IDENTIFIER(p)
    bare_number : UNSIGNED NUMBER_IDENTIFIER

LilyPondSyntacticalDefinition.p_bare_number__bare_number_closed(p)
    bare_number : bare_number_closed

LilyPondSyntacticalDefinition.p_bare_number_closed__NUMBER_IDENTIFIER(p)
    bare_number_closed : NUMBER_IDENTIFIER

LilyPondSyntacticalDefinition.p_bare_number_closed__REAL(p)
    bare_number_closed : REAL

LilyPondSyntacticalDefinition.p_bare_number_closed__UNSIGNED(p)
    bare_number_closed : UNSIGNED

LilyPondSyntacticalDefinition.p_bare_unsigned__UNSIGNED(p)
    bare_unsigned : UNSIGNED

LilyPondSyntacticalDefinition.p_braced_music_list__Chr123__music_list__Chr125(p)
    braced_music_list : '{' music_list '}'

LilyPondSyntacticalDefinition.p_chord_body__ANGLE_OPEN__chord_body_elements__ANGLE_CLOSE(p)
    chord_body : ANGLE_OPEN chord_body_elements ANGLE_CLOSE

LilyPondSyntacticalDefinition.p_chord_body_element__music_function_chord_body(p)
    chord_body_element : music_function_chord_body

LilyPondSyntacticalDefinition.p_chord_body_element__pitch_exclamations_questions_octave(p)
    chord_body_element : pitch exclamations questions octave_check post_events

LilyPondSyntacticalDefinition.p_chord_body_elements__Empty(p)
    chord_body_elements :

LilyPondSyntacticalDefinition.p_chord_body_elements__chord_body_elements__chord_body_element(p)
    chord_body_elements : chord_body_elements chord_body_element

LilyPondSyntacticalDefinition.p_closed_music__complex_music_prefix__closed_music(p)
    closed_music : complex_music_prefix closed_music

LilyPondSyntacticalDefinition.p_closed_music__music_bare(p)
    closed_music : music_bare

LilyPondSyntacticalDefinition.p_command_element__Chr124(p)
    command_element : '!'

LilyPondSyntacticalDefinition.p_command_element__E_BACKSLASH(p)
    command_element : E_BACKSLASH

LilyPondSyntacticalDefinition.p_command_element__command_event(p)
    command_element : command_event

```

```

LilyPondSyntacticalDefinition.p_command_event__tempo_event(p)
    command_event : tempo_event

LilyPondSyntacticalDefinition.p_complex_music__complex_music_prefix__music(p)
    complex_music : complex_music_prefix music

LilyPondSyntacticalDefinition.p_complex_music__music_function_call(p)
    complex_music : music_function_call

LilyPondSyntacticalDefinition.p_complex_music_prefix__CONTEXT__simple_string__optional_id__optional_context_mod(p)
    complex_music_prefix : CONTEXT simple_string optional_id optional_context_mod

LilyPondSyntacticalDefinition.p_complex_music_prefix__NEWCONTEXT__simple_string__optional_id__optional_context_mod(p)
    complex_music_prefix : NEWCONTEXT simple_string optional_id optional_context_mod

LilyPondSyntacticalDefinition.p_composite_music__complex_music(p)
    composite_music : complex_music

LilyPondSyntacticalDefinition.p_composite_music__music_bare(p)
    composite_music : music_bare

LilyPondSyntacticalDefinition.p_context_change__CHANGE__STRING__Chr61__STRING(p)
    context_change : CHANGE STRING '=' STRING

LilyPondSyntacticalDefinition.p_context_def_spec_block__CONTEXT__Chr123__context_def_spec_block__context_def_spec_body(p)
    context_def_spec_block : CONTEXT '{ 'context_def_spec_body '}'

LilyPondSyntacticalDefinition.p_context_def_spec_body__CONTEXT_DEF_IDENTIFIER(p)
    context_def_spec_body : CONTEXT_DEF_IDENTIFIER

LilyPondSyntacticalDefinition.p_context_def_spec_body__Empty(p)
    context_def_spec_body :

LilyPondSyntacticalDefinition.p_context_def_spec_body__context_def_spec_body__context_mod(p)
    context_def_spec_body : context_def_spec_body context_mod

LilyPondSyntacticalDefinition.p_context_def_spec_body__context_def_spec_body__context_modification(p)
    context_def_spec_body : context_def_spec_body context_modification

LilyPondSyntacticalDefinition.p_context_def_spec_body__context_def_spec_body__embedded_scm(p)
    context_def_spec_body : context_def_spec_body embedded_scm

LilyPondSyntacticalDefinition.p_context_mod__property_operation(p)
    context_mod : property_operation

LilyPondSyntacticalDefinition.p_context_mod_list__Empty(p)
    context_mod_list :

LilyPondSyntacticalDefinition.p_context_mod_list__context_mod_list__CONTEXT_MOD_IDENTIFIER(p)
    context_mod_list : context_mod_list CONTEXT_MOD_IDENTIFIER

LilyPondSyntacticalDefinition.p_context_mod_list__context_mod_list__context_mod(p)
    context_mod_list : context_mod_list context_mod

LilyPondSyntacticalDefinition.p_context_mod_list__context_mod_list__embedded_scm(p)
    context_mod_list : context_mod_list embedded_scm

LilyPondSyntacticalDefinition.p_context_modification__CONTEXT_MOD_IDENTIFIER(p)
    context_modification : CONTEXT_MOD_IDENTIFIER

LilyPondSyntacticalDefinition.p_context_modification__WITH__CONTEXT_MOD_IDENTIFIER(p)
    context_modification : WITH CONTEXT_MOD_IDENTIFIER

```

```

LilyPondSyntacticalDefinition.p_context_modification__WITH__Chr123__context_mod_list__Chr123
    context_modification : WITH '{ context_mod_list '}'

LilyPondSyntacticalDefinition.p_context_modification__WITH__embedded_scm_closed(p)
    context_modification : WITH embedded_scm_closed

LilyPondSyntacticalDefinition.p_context_prop_spec__simple_string(p)
    context_prop_spec : simple_string

LilyPondSyntacticalDefinition.p_context_prop_spec__simple_string__Chr46__simple_string(p)
    context_prop_spec : simple_string '.' simple_string

LilyPondSyntacticalDefinition.p_direction_less_char__Chr126(p)
    direction_less_char : '~'

LilyPondSyntacticalDefinition.p_direction_less_char__Chr40(p)
    direction_less_char : '('

LilyPondSyntacticalDefinition.p_direction_less_char__Chr41(p)
    direction_less_char : ')'

LilyPondSyntacticalDefinition.p_direction_less_char__Chr91(p)
    direction_less_char : '['

LilyPondSyntacticalDefinition.p_direction_less_char__Chr93(p)
    direction_less_char : ']'

LilyPondSyntacticalDefinition.p_direction_less_char__E_ANGLE_CLOSE(p)
    direction_less_char : E_ANGLE_CLOSE

LilyPondSyntacticalDefinition.p_direction_less_char__E_ANGLE_OPEN(p)
    direction_less_char : E_ANGLE_OPEN

LilyPondSyntacticalDefinition.p_direction_less_char__E_CLOSE(p)
    direction_less_char : E_CLOSE

LilyPondSyntacticalDefinition.p_direction_less_char__E_EXCLAMATION(p)
    direction_less_char : E_EXCLAMATION

LilyPondSyntacticalDefinition.p_direction_less_char__E_OPEN(p)
    direction_less_char : E_OPEN

LilyPondSyntacticalDefinition.p_direction_less_event__EVENT_IDENTIFIER(p)
    direction_less_event : EVENT_IDENTIFIER

LilyPondSyntacticalDefinition.p_direction_less_event__direction_less_char(p)
    direction_less_event : direction_less_char

LilyPondSyntacticalDefinition.p_direction_less_event__event_function_event(p)
    direction_less_event : event_function_event

LilyPondSyntacticalDefinition.p_direction_less_event__tremolo_type(p)
    direction_less_event : tremolo_type

LilyPondSyntacticalDefinition.p_direction_reqd_event__gen_text_def(p)
    direction_reqd_event : gen_text_def

LilyPondSyntacticalDefinition.p_direction_reqd_event__script_abbreviation(p)
    direction_reqd_event : script_abbreviation

LilyPondSyntacticalDefinition.p_dots__Empty(p)
    dots :

```

```

LilyPondSyntacticalDefinition.p_dots__dots__Chr46(p)
    dots : dots ‘.’

LilyPondSyntacticalDefinition.p_duration_length__multiplied_duration(p)
    duration_length : multiplied_duration

LilyPondSyntacticalDefinition.p_embedded_scm__embedded_scm_bare(p)
    embedded_scm : embedded_scm_bare

LilyPondSyntacticalDefinition.p_embedded_scm__scm_function_call(p)
    embedded_scm : scm_function_call

LilyPondSyntacticalDefinition.p_embedded_scm_arg__embedded_scm_bare_arg(p)
    embedded_scm_arg : embedded_scm_bare_arg

LilyPondSyntacticalDefinition.p_embedded_scm_arg__music_arg(p)
    embedded_scm_arg : music_arg

LilyPondSyntacticalDefinition.p_embedded_scm_arg__scm_function_call(p)
    embedded_scm_arg : scm_function_call

LilyPondSyntacticalDefinition.p_embedded_scm_arg_closed__closed_music(p)
    embedded_scm_arg_closed : closed_music

LilyPondSyntacticalDefinition.p_embedded_scm_arg_closed__embedded_scm_bare_arg(p)
    embedded_scm_arg_closed : embedded_scm_bare_arg

LilyPondSyntacticalDefinition.p_embedded_scm_arg_closed__scm_function_call_closed(p)
    embedded_scm_arg_closed : scm_function_call_closed

LilyPondSyntacticalDefinition.p_embedded_scm_bare__SCM_IDENTIFIER(p)
    embedded_scm_bare : SCM_IDENTIFIER

LilyPondSyntacticalDefinition.p_embedded_scm_bare__SCM_TOKEN(p)
    embedded_scm_bare : SCM_TOKEN

LilyPondSyntacticalDefinition.p_embedded_scm_bare_arg__STRING(p)
    embedded_scm_bare_arg : STRING

LilyPondSyntacticalDefinition.p_embedded_scm_bare_arg__STRING_IDENTIFIER(p)
    embedded_scm_bare_arg : STRING_IDENTIFIER

LilyPondSyntacticalDefinition.p_embedded_scm_bare_arg__context_def_spec_block(p)
    embedded_scm_bare_arg : context_def_spec_block

LilyPondSyntacticalDefinition.p_embedded_scm_bare_arg__context_modification(p)
    embedded_scm_bare_arg : context_modification

LilyPondSyntacticalDefinition.p_embedded_scm_bare_arg__embedded_scm_bare(p)
    embedded_scm_bare_arg : embedded_scm_bare

LilyPondSyntacticalDefinition.p_embedded_scm_bare_arg__full_markup(p)
    embedded_scm_bare_arg : full_markup

LilyPondSyntacticalDefinition.p_embedded_scm_bare_arg__full_markup_list(p)
    embedded_scm_bare_arg : full_markup_list

LilyPondSyntacticalDefinition.p_embedded_scm_bare_arg__output_def(p)
    embedded_scm_bare_arg : output_def

LilyPondSyntacticalDefinition.p_embedded_scm_bare_arg__score_block(p)
    embedded_scm_bare_arg : score_block

```

`LilyPondSyntacticalDefinition.p_embedded_scm_chord_body__SCM_FUNCTION__music_function_chord`
`embedded_scm_chord_body : SCM_FUNCTION music_function_chord_body_arglist`

`LilyPondSyntacticalDefinition.p_embedded_scm_chord_body__bare_number(p)`
`embedded_scm_chord_body : bare_number`

`LilyPondSyntacticalDefinition.p_embedded_scm_chord_body__chord_body_element(p)`
`embedded_scm_chord_body : chord_body_element`

`LilyPondSyntacticalDefinition.p_embedded_scm_chord_body__embedded_scm_bare_arg(p)`
`embedded_scm_chord_body : embedded_scm_bare_arg`

`LilyPondSyntacticalDefinition.p_embedded_scm_chord_body__fraction(p)`
`embedded_scm_chord_body : fraction`

`LilyPondSyntacticalDefinition.p_embedded_scm_closed__embedded_scm_bare(p)`
`embedded_scm_closed : embedded_scm_bare`

`LilyPondSyntacticalDefinition.p_embedded_scm_closed__scm_function_call_closed(p)`
`embedded_scm_closed : scm_function_call_closed`

`LilyPondSyntacticalDefinition.p_error(p)`

`LilyPondSyntacticalDefinition.p_event_chord__CHORD_REPETITION__optional_notemode_duration`
`event_chord : CHORD_REPETITION optional_notemode_duration post_events`

`LilyPondSyntacticalDefinition.p_event_chord__MULTI_MEASURE_REST__optional_notemode_duration`
`event_chord : MULTI_MEASURE_REST optional_notemode_duration post_events`

`LilyPondSyntacticalDefinition.p_event_chord__command_element(p)`
`event_chord : command_element`

`LilyPondSyntacticalDefinition.p_event_chord__note_chord_element(p)`
`event_chord : note_chord_element`

`LilyPondSyntacticalDefinition.p_event_chord__simple_chord_elements__post_events(p)`
`event_chord : simple_chord_elements post_events`

`LilyPondSyntacticalDefinition.p_event_function_event__EVENT_FUNCTION__function_arglist_closed`
`event_function_event : EVENT_FUNCTION function_arglist_closed`

`LilyPondSyntacticalDefinition.p_exclamations__Empty(p)`
`exclamations :`

`LilyPondSyntacticalDefinition.p_exclamations__exclamations__Chr33(p)`
`exclamations : exclamations ‘!’`

`LilyPondSyntacticalDefinition.p_fingering__UNSIGNED(p)`
`fingering : UNSIGNED`

`LilyPondSyntacticalDefinition.p_fraction__FRACTION(p)`
`fraction : FRACTION`

`LilyPondSyntacticalDefinition.p_fraction__UNSIGNED__Chr47__UNSIGNED(p)`
`fraction : UNSIGNED ‘/’ UNSIGNED`

`LilyPondSyntacticalDefinition.p_full_markup__MARKUP_IDENTIFIER(p)`
`full_markup : MARKUP_IDENTIFIER`

`LilyPondSyntacticalDefinition.p_full_markup__MARKUP__markup_top(p)`
`full_markup : MARKUP markup_top`

`LilyPondSyntacticalDefinition.p_full_markup_list__MARKUPLIST_IDENTIFIER(p)`
`full_markup_list : MARKUPLIST_IDENTIFIER`

LilyPondSyntacticalDefinition.**p_full_markup_list__MARKUPLIST__markup_list**(*p*)
 full_markup_list : MARKUPLIST markup_list

LilyPondSyntacticalDefinition.**p_function_arglist__function_arglist_common**(*p*)
 function_arglist : function_arglist_common

LilyPondSyntacticalDefinition.**p_function_arglist__function_arglist_nonbackup**(*p*)
 function_arglist : function_arglist_nonbackup

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_DURATION__**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_DURATION function_arglist_closed_keep dura-
 tion_length

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_PITCH__fu**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_PITCH function_arglist_keep
 pitch_also_in_chords

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_backup BACKUP

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed_keep '-' NUM-
 BER_IDENTIFIER

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed_keep '-' REAL

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed_keep '-' UN-
 SIGNED

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed_keep FRACTION

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed_keep NUM-
 BER_IDENTIFIER

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed_keep REAL

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed_keep UNSIGNED

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed_keep
 post_event_nofinger

LilyPondSyntacticalDefinition.**p_function_arglist_backup__EXPECT_OPTIONAL__EXPECT_SCM__funct**
 function_arglist_backup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_keep embed-
 ded_scm_arg_closed

LilyPondSyntacticalDefinition.**p_function_arglist_backup__function_arglist_backup__REPARSE__**
 function_arglist_backup : function_arglist_backup REPARSE bare_number

LilyPondSyntacticalDefinition.**p_function_arglist_backup__function_arglist_backup__REPARSE__**
 function_arglist_backup : function_arglist_backup REPARSE embedded_scm_arg_closed

LilyPondSyntacticalDefinition.**p_function_arglist_backup__function_arglist_backup__REPARSE__**
 function_arglist_backup : function_arglist_backup REPARSE fraction

LilyPondSyntacticalDefinition.**p_function_arglist_bare__EXPECT_DURATION__function_arglist_c**
 function_arglist_bare : EXPECT_DURATION function_arglist_closed_optional duration_length

```

LilyPondSyntacticalDefinition.p_function_arglist_bare_EXPECT_NO_MORE_ARGS (p)
    function_arglist_bare : EXPECT_NO_MORE_ARGS

LilyPondSyntacticalDefinition.p_function_arglist_bare_EXPECT_OPTIONAL_EXPECT_DURATION__f
    function_arglist_bare : EXPECT_OPTIONAL EXPECT_DURATION function_arglist_skip DEFAULT

LilyPondSyntacticalDefinition.p_function_arglist_bare_EXPECT_OPTIONAL_EXPECT_PITCH__func
    function_arglist_bare : EXPECT_OPTIONAL EXPECT_PITCH function_arglist_skip DEFAULT

LilyPondSyntacticalDefinition.p_function_arglist_bare_EXPECT_OPTIONAL_EXPECT_SCM__functio
    function_arglist_bare : EXPECT_OPTIONAL EXPECT_SCM function_arglist_skip DEFAULT

LilyPondSyntacticalDefinition.p_function_arglist_bare_EXPECT_PITCH__function_arglist_opti
    function_arglist_bare : EXPECT_PITCH function_arglist_optional pitch_also_in_chords

LilyPondSyntacticalDefinition.p_function_arglist_closed__function_arglist_closed_common (p)
    function_arglist_closed : function_arglist_closed_common

LilyPondSyntacticalDefinition.p_function_arglist_closed__function_arglist_nonbackup (p)
    function_arglist_closed : function_arglist_nonbackup

LilyPondSyntacticalDefinition.p_function_arglist_closed_common_EXPECT_SCM__function_arglis
    function_arglist_closed_common : EXPECT_SCM function_arglist_closed_optional '-' NUM-
    BER_IDENTIFIER

LilyPondSyntacticalDefinition.p_function_arglist_closed_common_EXPECT_SCM__function_arglis
    function_arglist_closed_common : EXPECT_SCM function_arglist_closed_optional '-' REAL

LilyPondSyntacticalDefinition.p_function_arglist_closed_common_EXPECT_SCM__function_arglis
    function_arglist_closed_common : EXPECT_SCM function_arglist_closed_optional '-' UNSIGNED

LilyPondSyntacticalDefinition.p_function_arglist_closed_common_EXPECT_SCM__function_arglis
    function_arglist_closed_common : EXPECT_SCM function_arglist_closed_optional bare_number

LilyPondSyntacticalDefinition.p_function_arglist_closed_common_EXPECT_SCM__function_arglis
    function_arglist_closed_common : EXPECT_SCM function_arglist_closed_optional fraction

LilyPondSyntacticalDefinition.p_function_arglist_closed_common_EXPECT_SCM__function_arglis
    function_arglist_closed_common : EXPECT_SCM function_arglist_closed_optional post_event_nofinger

LilyPondSyntacticalDefinition.p_function_arglist_closed_common_EXPECT_SCM__function_arglis
    function_arglist_closed_common : EXPECT_SCM function_arglist_optional embedded_scm_arg_closed

LilyPondSyntacticalDefinition.p_function_arglist_closed_common__function_arglist_bare (p)
    function_arglist_closed_common : function_arglist_bare

LilyPondSyntacticalDefinition.p_function_arglist_closed_keep__function_arglist_backup (p)
    function_arglist_closed_keep : function_arglist_backup

LilyPondSyntacticalDefinition.p_function_arglist_closed_keep__function_arglist_closed_comme
    function_arglist_closed_keep : function_arglist_closed_common

LilyPondSyntacticalDefinition.p_function_arglist_closed_optional_EXPECT_OPTIONAL_EXPECT_P
    function_arglist_closed_optional : EXPECT_OPTIONAL EXPECT_DURATION func-
    tion_arglist_closed_optional

LilyPondSyntacticalDefinition.p_function_arglist_closed_optional_EXPECT_OPTIONAL_EXPECT_P
    function_arglist_closed_optional : EXPECT_OPTIONAL EXPECT_PITCH function_arglist_closed_optional

LilyPondSyntacticalDefinition.p_function_arglist_closed_optional__function_arglist_backup
    function_arglist_closed_optional : function_arglist_backup BACKUP

LilyPondSyntacticalDefinition.p_function_arglist_closed_optional__function_arglist_closed_l
    function_arglist_closed_optional : function_arglist_closed_keep %prec FUNCTION_ARGLIST

```

```

LilyPondSyntacticalDefinition.p_function_arglist_common_EXPECT_SCM_function_arglist_closed_optional_bare_number
    function_arglist_common : EXPECT_SCM function_arglist_closed_optional bare_number

LilyPondSyntacticalDefinition.p_function_arglist_common_EXPECT_SCM_function_arglist_closed_optional_fraction
    function_arglist_common : EXPECT_SCM function_arglist_closed_optional fraction

LilyPondSyntacticalDefinition.p_function_arglist_common_EXPECT_SCM_function_arglist_closed_optional_post_event_nofinger
    function_arglist_common : EXPECT_SCM function_arglist_closed_optional post_event_nofinger

LilyPondSyntacticalDefinition.p_function_arglist_common_EXPECT_SCM_function_arglist_optional_embedded_scm_arg
    function_arglist_common : EXPECT_SCM function_arglist_optional embedded_scm_arg

LilyPondSyntacticalDefinition.p_function_arglist_common_function_arglist_bare(p)
    function_arglist_common : function_arglist_bare

LilyPondSyntacticalDefinition.p_function_arglist_common_function_arglist_common_minus(p)
    function_arglist_common : function_arglist_common_minus

LilyPondSyntacticalDefinition.p_function_arglist_common_minus_EXPECT_SCM_function_arglist_closed_optional_NUM-BER_IDENTIFIER
    function_arglist_common_minus : EXPECT_SCM function_arglist_closed_optional '-' NUM-
    BER_IDENTIFIER

LilyPondSyntacticalDefinition.p_function_arglist_common_minus_EXPECT_SCM_function_arglist_closed_optional_REAL
    function_arglist_common_minus : EXPECT_SCM function_arglist_closed_optional '-' REAL

LilyPondSyntacticalDefinition.p_function_arglist_common_minus_EXPECT_SCM_function_arglist_closed_optional_UNSIGNED
    function_arglist_common_minus : EXPECT_SCM function_arglist_closed_optional '-' UNSIGNED

LilyPondSyntacticalDefinition.p_function_arglist_common_minus_function_arglist_common_minus_REPARSE_bare_number
    function_arglist_common_minus : function_arglist_common_minus REPARSE bare_number

LilyPondSyntacticalDefinition.p_function_arglist_keep_function_arglist_backup(p)
    function_arglist_keep : function_arglist_backup

LilyPondSyntacticalDefinition.p_function_arglist_keep_function_arglist_common(p)
    function_arglist_keep : function_arglist_common

LilyPondSyntacticalDefinition.p_function_arglist_nonbackup_EXPECT_OPTIONAL_EXPECT_DURATION_function_arglist_closed_duration_length
    function_arglist_nonbackup : EXPECT_OPTIONAL EXPECT_DURATION function_arglist_closed dura-
    tion_length

LilyPondSyntacticalDefinition.p_function_arglist_nonbackup_EXPECT_OPTIONAL_EXPECT_PITCH_function_arglist_pitch_also_in_chords
    function_arglist_nonbackup : EXPECT_OPTIONAL EXPECT_PITCH function_arglist pitch_also_in_chords

LilyPondSyntacticalDefinition.p_function_arglist_nonbackup_EXPECT_OPTIONAL_EXPECT_SCM_function_arglist_embedded_scm_arg_closed
    function_arglist_nonbackup : EXPECT_OPTIONAL EXPECT_SCM function_arglist embed-
    ded_scm_arg_closed

LilyPondSyntacticalDefinition.p_function_arglist_nonbackup_EXPECT_OPTIONAL_EXPECT_SCM_function_arglist_closed_NUM-BER_IDENTIFIER
    function_arglist_nonbackup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed '-' NUM-
    BER_IDENTIFIER

LilyPondSyntacticalDefinition.p_function_arglist_nonbackup_EXPECT_OPTIONAL_EXPECT_SCM_function_arglist_closed_REAL
    function_arglist_nonbackup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed '-' REAL

LilyPondSyntacticalDefinition.p_function_arglist_nonbackup_EXPECT_OPTIONAL_EXPECT_SCM_function_arglist_closed_UNSIGNED
    function_arglist_nonbackup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed '-' UNSIGNED

LilyPondSyntacticalDefinition.p_function_arglist_nonbackup_EXPECT_OPTIONAL_EXPECT_SCM_function_arglist_closed_FRACTION
    function_arglist_nonbackup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed FRACTION

```



```

LilyPondSyntacticalDefinition.p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_SCM__f
    function_arglist_nonbackup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed
    bare_number_closed

LilyPondSyntacticalDefinition.p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_SCM__f
    function_arglist_nonbackup : EXPECT_OPTIONAL EXPECT_SCM function_arglist_closed
    post_event_nofinger

LilyPondSyntacticalDefinition.p_function_arglist_optional__EXPECT_OPTIONAL__EXPECT_DURATION
    function_arglist_optional : EXPECT_OPTIONAL EXPECT_DURATION function_arglist_optional

LilyPondSyntacticalDefinition.p_function_arglist_optional__EXPECT_OPTIONAL__EXPECT_PITCH__f
    function_arglist_optional : EXPECT_OPTIONAL EXPECT_PITCH function_arglist_optional

LilyPondSyntacticalDefinition.p_function_arglist_optional__function_arglist_backup__BACKUP
    function_arglist_optional : function_arglist_backup BACKUP

LilyPondSyntacticalDefinition.p_function_arglist_optional__function_arglist_keep(p)
    function_arglist_optional : function_arglist_keep %prec FUNCTION_ARGLIST

LilyPondSyntacticalDefinition.p_function_arglist_skip__EXPECT_OPTIONAL__EXPECT_DURATION__f
    function_arglist_skip : EXPECT_OPTIONAL EXPECT_DURATION function_arglist_skip %prec FUNC-
    TION_ARGLIST

LilyPondSyntacticalDefinition.p_function_arglist_skip__EXPECT_OPTIONAL__EXPECT_PITCH__func
    function_arglist_skip : EXPECT_OPTIONAL EXPECT_PITCH function_arglist_skip %prec FUNC-
    TION_ARGLIST

LilyPondSyntacticalDefinition.p_function_arglist_skip__EXPECT_OPTIONAL__EXPECT_SCM__functio
    function_arglist_skip : EXPECT_OPTIONAL EXPECT_SCM function_arglist_skip %prec FUNC-
    TION_ARGLIST

LilyPondSyntacticalDefinition.p_function_arglist_skip__function_arglist_common(p)
    function_arglist_skip : function_arglist_common

LilyPondSyntacticalDefinition.p_gen_text_def__full_markup(p)
    gen_text_def : full_markup

LilyPondSyntacticalDefinition.p_gen_text_def__simple_string(p)
    gen_text_def : simple_string

LilyPondSyntacticalDefinition.p_grouped_music_list__sequential_music(p)
    grouped_music_list : sequential_music

LilyPondSyntacticalDefinition.p_grouped_music_list__simultaneous_music(p)
    grouped_music_list : simultaneous_music

LilyPondSyntacticalDefinition.p_identifier_init__context_def_spec_block(p)
    identifier_init : context_def_spec_block

LilyPondSyntacticalDefinition.p_identifier_init__context_modification(p)
    identifier_init : context_modification

LilyPondSyntacticalDefinition.p_identifier_init__embedded_scm(p)
    identifier_init : embedded_scm

LilyPondSyntacticalDefinition.p_identifier_init__full_markup(p)
    identifier_init : full_markup

LilyPondSyntacticalDefinition.p_identifier_init__full_markup_list(p)
    identifier_init : full_markup_list

```

```

LilyPondSyntacticalDefinition.p_identifier_init__music(p)
    identifier_init : music

LilyPondSyntacticalDefinition.p_identifier_init__number_expression(p)
    identifier_init : number_expression

LilyPondSyntacticalDefinition.p_identifier_init__output_def(p)
    identifier_init : output_def

LilyPondSyntacticalDefinition.p_identifier_init__post_event_nofinger(p)
    identifier_init : post_event_nofinger

LilyPondSyntacticalDefinition.p_identifier_init__score_block(p)
    identifier_init : score_block

LilyPondSyntacticalDefinition.p_identifier_init__string(p)
    identifier_init : string

LilyPondSyntacticalDefinition.p_lilypond__Empty(p)
    lilypond :

LilyPondSyntacticalDefinition.p_lilypond__lilypond__assignment(p)
    lilypond : lilypond assignment

LilyPondSyntacticalDefinition.p_lilypond__lilypond__error(p)
    lilypond : lilypond error

LilyPondSyntacticalDefinition.p_lilypond__lilypond__toplevel_expression(p)
    lilypond : lilypond toplevel_expression

LilyPondSyntacticalDefinition.p_lilypond_header__HEADER__Chr123__lilypond_header_body__Chr123
    lilypond_header : HEADER '{ lilypond_header_body }'

LilyPondSyntacticalDefinition.p_lilypond_header_body__Empty(p)
    lilypond_header_body :

LilyPondSyntacticalDefinition.p_lilypond_header_body__lilypond_header_body__assignment(p)
    lilypond_header_body : lilypond_header_body assignment

LilyPondSyntacticalDefinition.p_markup__markup_head_1_list__simple_markup(p)
    markup : markup_head_1_list simple_markup

LilyPondSyntacticalDefinition.p_markup__simple_markup(p)
    markup : simple_markup

LilyPondSyntacticalDefinition.p_markup_braced_list__Chr123__markup_braced_list_body__Chr123
    markup_braced_list : '{ markup_braced_list_body }'

LilyPondSyntacticalDefinition.p_markup_braced_list_body__Empty(p)
    markup_braced_list_body :

LilyPondSyntacticalDefinition.p_markup_braced_list_body__markup_braced_list_body__markup(p)
    markup_braced_list_body : markup_braced_list_body markup

LilyPondSyntacticalDefinition.p_markup_braced_list_body__markup_braced_list_body__markup_list
    markup_braced_list_body : markup_braced_list_body markup_list

LilyPondSyntacticalDefinition.p_markup_command_basic_arguments__EXPECT_MARKUP_LIST__markup
    markup_command_basic_arguments : EXPECT_MARKUP_LIST markup_command_list_arguments
    markup_list

LilyPondSyntacticalDefinition.p_markup_command_basic_arguments__EXPECT_NO_MORE_ARGS(p)
    markup_command_basic_arguments : EXPECT_NO_MORE_ARGS

```

```

LilyPondSyntacticalDefinition.p_markup_command_basic_arguments_EXPECT_SCM_markup_command
markup_command_basic_arguments : EXPECT_SCM markup_command_list_arguments embed-
ded_scm_closed

LilyPondSyntacticalDefinition.p_markup_command_list_MARKUP_LIST_FUNCTION_markup_command_
markup_command_list : MARKUP_LIST_FUNCTION markup_command_list_arguments

LilyPondSyntacticalDefinition.p_markup_command_list_arguments_EXPECT_MARKUP_markup_comma
markup_command_list_arguments : EXPECT_MARKUP markup_command_list_arguments markup

LilyPondSyntacticalDefinition.p_markup_command_list_arguments_markup_command_basic_argumen
markup_command_list_arguments : markup_command_basic_arguments

LilyPondSyntacticalDefinition.p_markup_composed_list_markup_head_1_list_markup_braced_li
markup_composed_list : markup_head_1_list markup_braced_list

LilyPondSyntacticalDefinition.p_markup_head_1_item_MARKUP_FUNCTION_EXPECT_MARKUP_markup_
markup_head_1_item : MARKUP_FUNCTION EXPECT_MARKUP markup_command_list_arguments

LilyPondSyntacticalDefinition.p_markup_head_1_list_markup_head_1_item(p)
markup_head_1_list : markup_head_1_item

LilyPondSyntacticalDefinition.p_markup_head_1_list_markup_head_1_list_markup_head_1_item
markup_head_1_list : markup_head_1_list markup_head_1_item

LilyPondSyntacticalDefinition.p_markup_list_MARKUPLIST_IDENTIFIER(p)
markup_list : MARKUPLIST_IDENTIFIER

LilyPondSyntacticalDefinition.p_markup_list_markup_braced_list(p)
markup_list : markup_braced_list

LilyPondSyntacticalDefinition.p_markup_list_markup_command_list(p)
markup_list : markup_command_list

LilyPondSyntacticalDefinition.p_markup_list_markup_composed_list(p)
markup_list : markup_composed_list

LilyPondSyntacticalDefinition.p_markup_list_markup_scm_MARKUPLIST_IDENTIFIER(p)
markup_list : markup_scm MARKUPLIST_IDENTIFIER

LilyPondSyntacticalDefinition.p_markup_scm_embedded_scm_bare_BACKUP(p)
markup_scm : embedded_scm_bare BACKUP

LilyPondSyntacticalDefinition.p_markup_top_markup_head_1_list_simple_markup(p)
markup_top : markup_head_1_list simple_markup

LilyPondSyntacticalDefinition.p_markup_top_markup_list(p)
markup_top : markup_list

LilyPondSyntacticalDefinition.p_markup_top_simple_markup(p)
markup_top : simple_markup

LilyPondSyntacticalDefinition.p_multiplied_duration_multiplied_duration_Chr42_FRACTION(p)
multiplied_duration : multiplied_duration "*" FRACTION

LilyPondSyntacticalDefinition.p_multiplied_duration_multiplied_duration_Chr42_bare_unsig
multiplied_duration : multiplied_duration "*" bare_unsigned

LilyPondSyntacticalDefinition.p_multiplied_duration_steno_duration(p)
multiplied_duration : steno_duration

LilyPondSyntacticalDefinition.p_music_composite_music(p)
music : composite_music %prec COMPOSITE

```

```

LilyPondSyntacticalDefinition.p_music__simple_music(p)
    music : simple_music

LilyPondSyntacticalDefinition.p_music_arg__composite_music(p)
    music_arg : composite_music %prec COMPOSITE

LilyPondSyntacticalDefinition.p_music_arg__simple_music(p)
    music_arg : simple_music

LilyPondSyntacticalDefinition.p_music_bare__MUSIC_IDENTIFIER(p)
    music_bare : MUSIC_IDENTIFIER

LilyPondSyntacticalDefinition.p_music_bare__grouped_music_list(p)
    music_bare : grouped_music_list

LilyPondSyntacticalDefinition.p_music_function_call__MUSIC_FUNCTION__function_arglist(p)
    music_function_call : MUSIC_FUNCTION function_arglist

LilyPondSyntacticalDefinition.p_music_function_chord_body__MUSIC_FUNCTION__music_function_chord_body_arglist(p)
    music_function_chord_body : MUSIC_FUNCTION music_function_chord_body_arglist

LilyPondSyntacticalDefinition.p_music_function_chord_body_arglist__EXPECT_SCM__music_function_chord_body_arglist(p)
    music_function_chord_body_arglist : EXPECT_SCM music_function_chord_body_arglist embed-
    ded_scm_chord_body

LilyPondSyntacticalDefinition.p_music_function_chord_body_arglist__function_arglist_bare(p)
    music_function_chord_body_arglist : function_arglist_bare

LilyPondSyntacticalDefinition.p_music_function_event__MUSIC_FUNCTION__function_arglist_closed(p)
    music_function_event : MUSIC_FUNCTION function_arglist_closed

LilyPondSyntacticalDefinition.p_music_list__Empty(p)
    music_list :

LilyPondSyntacticalDefinition.p_music_list__music_list__embedded_scm(p)
    music_list : music_list embedded_scm

LilyPondSyntacticalDefinition.p_music_list__music_list__error(p)
    music_list : music_list error

LilyPondSyntacticalDefinition.p_music_list__music_list__music(p)
    music_list : music_list music

LilyPondSyntacticalDefinition.p_music_property_def__simple_music_property_def(p)
    music_property_def : simple_music_property_def

LilyPondSyntacticalDefinition.p_note_chord_element__chord_body__optional_notemode_duration__post_events(p)
    note_chord_element : chord_body optional_notemode_duration post_events

LilyPondSyntacticalDefinition.p_number_expression__number_expression__Chr43__number_term(p)
    number_expression : number_expression '+' number_term

LilyPondSyntacticalDefinition.p_number_expression__number_expression__Chr45__number_term(p)
    number_expression : number_expression '-' number_term

LilyPondSyntacticalDefinition.p_number_expression__number_term(p)
    number_expression : number_term

LilyPondSyntacticalDefinition.p_number_factor__Chr45__number_factor(p)
    number_factor : '-' number_factor

LilyPondSyntacticalDefinition.p_number_factor__bare_number(p)
    number_factor : bare_number

```

```

LilyPondSyntacticalDefinition.p_number_term__number_factor(p)
    number_term : number_factor

LilyPondSyntacticalDefinition.p_number_term__number_factor__Chr42__number_factor(p)
    number_term : number_factor '*' number_factor

LilyPondSyntacticalDefinition.p_number_term__number_factor__Chr47__number_factor(p)
    number_term : number_factor '/' number_factor

LilyPondSyntacticalDefinition.p_octave_check__Chr61(p)
    octave_check : '='

LilyPondSyntacticalDefinition.p_octave_check__Chr61__sub_quotes(p)
    octave_check : '=' sub_quotes

LilyPondSyntacticalDefinition.p_octave_check__Chr61__sup_quotes(p)
    octave_check : '=' sup_quotes

LilyPondSyntacticalDefinition.p_octave_check__Empty(p)
    octave_check :

LilyPondSyntacticalDefinition.p_optional_context_mod__Empty(p)
    optional_context_mod :

LilyPondSyntacticalDefinition.p_optional_context_mod__context_modification(p)
    optional_context_mod : context_modification

LilyPondSyntacticalDefinition.p_optional_id__Chr61__simple_string(p)
    optional_id : '=' simple_string

LilyPondSyntacticalDefinition.p_optional_id__Empty(p)
    optional_id :

LilyPondSyntacticalDefinition.p_optional_notemode_duration__Empty(p)
    optional_notemode_duration :

LilyPondSyntacticalDefinition.p_optional_notemode_duration__multiplied_duration(p)
    optional_notemode_duration : multiplied_duration

LilyPondSyntacticalDefinition.p_optional_rest__Empty(p)
    optional_rest :

LilyPondSyntacticalDefinition.p_optional_rest__REST(p)
    optional_rest : REST

LilyPondSyntacticalDefinition.p_output_def__output_def_body__Chr125(p)
    output_def : output_def_body '}'

LilyPondSyntacticalDefinition.p_output_def_body__output_def_body__assignment(p)
    output_def_body : output_def_body assignment

LilyPondSyntacticalDefinition.p_output_def_body__output_def_head_with_mode_switch__Chr123(p)
    output_def_body : output_def_head_with_mode_switch '{'

LilyPondSyntacticalDefinition.p_output_def_body__output_def_head_with_mode_switch__Chr123(p)
    output_def_body : output_def_head_with_mode_switch '{' OUTPUT_DEF_IDENTIFIER

LilyPondSyntacticalDefinition.p_output_def_head__LAYOUT(p)
    output_def_head : LAYOUT

LilyPondSyntacticalDefinition.p_output_def_head__MIDI(p)
    output_def_head : MIDI

```

LilyPondSyntacticalDefinition.**p_output_def_head_PAPER**(*p*)
 output_def_head : PAPER

LilyPondSyntacticalDefinition.**p_output_def_head_with_mode_switch__output_def_head**(*p*)
 output_def_head_with_mode_switch : output_def_head

LilyPondSyntacticalDefinition.**p_pitch_PITCH_IDENTIFIER**(*p*)
 pitch : PITCH_IDENTIFIER

LilyPondSyntacticalDefinition.**p_pitch__steno_pitch**(*p*)
 pitch : steno_pitch

LilyPondSyntacticalDefinition.**p_pitch_also_in_chords__pitch**(*p*)
 pitch_also_in_chords : pitch

LilyPondSyntacticalDefinition.**p_pitch_also_in_chords__steno_tonic_pitch**(*p*)
 pitch_also_in_chords : steno_tonic_pitch

LilyPondSyntacticalDefinition.**p_post_event__Chr45__fingering**(*p*)
 post_event : ‘-’ fingering

LilyPondSyntacticalDefinition.**p_post_event__post_event_nofinger**(*p*)
 post_event : post_event_nofinger

LilyPondSyntacticalDefinition.**p_post_event_nofinger__Chr94__fingering**(*p*)
 post_event_nofinger : ‘^’ fingering

LilyPondSyntacticalDefinition.**p_post_event_nofinger__Chr95__fingering**(*p*)
 post_event_nofinger : ‘_’ fingering

LilyPondSyntacticalDefinition.**p_post_event_nofinger__EXTENDER**(*p*)
 post_event_nofinger : EXTENDER

LilyPondSyntacticalDefinition.**p_post_event_nofinger__HYPHEN**(*p*)
 post_event_nofinger : HYPHEN

LilyPondSyntacticalDefinition.**p_post_event_nofinger__direction_less_event**(*p*)
 post_event_nofinger : direction_less_event

LilyPondSyntacticalDefinition.**p_post_event_nofinger__script_dir__direction_less_event**(*p*)
 post_event_nofinger : script_dir direction_less_event

LilyPondSyntacticalDefinition.**p_post_event_nofinger__script_dir__direction_reqd_event**(*p*)
 post_event_nofinger : script_dir direction_reqd_event

LilyPondSyntacticalDefinition.**p_post_event_nofinger__script_dir__music_function_event**(*p*)
 post_event_nofinger : script_dir music_function_event

LilyPondSyntacticalDefinition.**p_post_event_nofinger__string_number_event**(*p*)
 post_event_nofinger : string_number_event

LilyPondSyntacticalDefinition.**p_post_events__Empty**(*p*)
 post_events :

LilyPondSyntacticalDefinition.**p_post_events__post_events__post_event**(*p*)
 post_events : post_events post_event

LilyPondSyntacticalDefinition.**p_property_operation__OVERRIDE__simple_string__property_path**
 property_operation : OVERRIDE simple_string property_path ‘=’ scalar

LilyPondSyntacticalDefinition.**p_property_operation__REVERT__simple_string__embedded_scm**(*p*)
 property_operation : REVERT simple_string embedded_scm

LilyPondSyntacticalDefinition.**p_property_operation__STRING__Chr61__scalar**(*p*)
property_operation : STRING ‘=’ scalar

LilyPondSyntacticalDefinition.**p_property_operation__UNSET__simple_string**(*p*)
property_operation : UNSET simple_string

LilyPondSyntacticalDefinition.**p_property_path__property_path_revved**(*p*)
property_path : property_path_revved

LilyPondSyntacticalDefinition.**p_property_path_revved__embedded_scm_closed**(*p*)
property_path_revved : embedded_scm_closed

LilyPondSyntacticalDefinition.**p_property_path_revved__property_path_revved__embedded_scm_c**(*p*)
property_path_revved : property_path_revved embedded_scm_closed

LilyPondSyntacticalDefinition.**p_questions__Empty**(*p*)
questions :

LilyPondSyntacticalDefinition.**p_questions__questions__Chr63**(*p*)
questions : questions ‘?’

LilyPondSyntacticalDefinition.**p_scalar__bare_number**(*p*)
scalar : bare_number

LilyPondSyntacticalDefinition.**p_scalar__embedded_scm_arg**(*p*)
scalar : embedded_scm_arg

LilyPondSyntacticalDefinition.**p_scalar_closed__bare_number**(*p*)
scalar_closed : bare_number

LilyPondSyntacticalDefinition.**p_scalar_closed__embedded_scm_arg_closed**(*p*)
scalar_closed : embedded_scm_arg_closed

LilyPondSyntacticalDefinition.**p_scm_function_call__SCM_FUNCTION__function_arglist**(*p*)
scm_function_call : SCM_FUNCTION function_arglist

LilyPondSyntacticalDefinition.**p_scm_function_call_closed__SCM_FUNCTION__function_arglist_c**(*p*)
scm_function_call_closed : SCM_FUNCTION function_arglist_closed %prec FUNCTION_ARGLIST

LilyPondSyntacticalDefinition.**p_score_block__SCORE__Chr123__score_body__Chr125**(*p*)
score_block : SCORE ‘{’ score_body ‘}’

LilyPondSyntacticalDefinition.**p_score_body__SCORE_IDENTIFIER**(*p*)
score_body : SCORE_IDENTIFIER

LilyPondSyntacticalDefinition.**p_score_body__music**(*p*)
score_body : music

LilyPondSyntacticalDefinition.**p_score_body__score_body__lilypond_header**(*p*)
score_body : score_body lilypond_header

LilyPondSyntacticalDefinition.**p_score_body__score_body__output_def**(*p*)
score_body : score_body output_def

LilyPondSyntacticalDefinition.**p_script_abbreviation__ANGLE_CLOSE**(*p*)
script_abbreviation : ANGLE_CLOSE

LilyPondSyntacticalDefinition.**p_script_abbreviation__Chr124**(*p*)
script_abbreviation : ‘l’

LilyPondSyntacticalDefinition.**p_script_abbreviation__Chr43**(*p*)
script_abbreviation : ‘+’

```

LilyPondSyntacticalDefinition.p_script_abbreviation__Chr45 (p)
    script_abbreviation : '-'

LilyPondSyntacticalDefinition.p_script_abbreviation__Chr46 (p)
    script_abbreviation : '.'

LilyPondSyntacticalDefinition.p_script_abbreviation__Chr94 (p)
    script_abbreviation : '^'

LilyPondSyntacticalDefinition.p_script_abbreviation__Chr95 (p)
    script_abbreviation : '_'

LilyPondSyntacticalDefinition.p_script_dir__Chr45 (p)
    script_dir : '-'

LilyPondSyntacticalDefinition.p_script_dir__Chr94 (p)
    script_dir : '^'

LilyPondSyntacticalDefinition.p_script_dir__Chr95 (p)
    script_dir : '_'

LilyPondSyntacticalDefinition.p_sequential_music__SEQUENTIAL__braced_music_list (p)
    sequential_music : SEQUENTIAL braced_music_list

LilyPondSyntacticalDefinition.p_sequential_music__braced_music_list (p)
    sequential_music : braced_music_list

LilyPondSyntacticalDefinition.p_simple_chord_elements__simple_element (p)
    simple_chord_elements : simple_element

LilyPondSyntacticalDefinition.p_simple_element__RESTNAME__optional_notemode_duration (p)
    simple_element : RESTNAME optional_notemode_duration

LilyPondSyntacticalDefinition.p_simple_element__pitch__exclamations__questions__octave_check__optional_notemode_duration__optional_rest (p)
    simple_element : pitch exclamations questions octave_check optional_notemode_duration optional_rest

LilyPondSyntacticalDefinition.p_simple_markup__MARKUP_FUNCTION__markup_command_basic_arguments (p)
    simple_markup : MARKUP_FUNCTION markup_command_basic_arguments

LilyPondSyntacticalDefinition.p_simple_markup__MARKUP_IDENTIFIER (p)
    simple_markup : MARKUP_IDENTIFIER

LilyPondSyntacticalDefinition.p_simple_markup__SCORE__Chr123__score_body__Chr125 (p)
    simple_markup : SCORE '{ score_body '}'

LilyPondSyntacticalDefinition.p_simple_markup__STRING (p)
    simple_markup : STRING

LilyPondSyntacticalDefinition.p_simple_markup__STRING_IDENTIFIER (p)
    simple_markup : STRING_IDENTIFIER

LilyPondSyntacticalDefinition.p_simple_markup__markup_scm__MARKUP_IDENTIFIER (p)
    simple_markup : markup_scm MARKUP_IDENTIFIER

LilyPondSyntacticalDefinition.p_simple_music__context_change (p)
    simple_music : context_change

LilyPondSyntacticalDefinition.p_simple_music__event_chord (p)
    simple_music : event_chord

LilyPondSyntacticalDefinition.p_simple_music__music_property_def (p)
    simple_music : music_property_def

```



```

LilyPondSyntacticalDefinition.p_simple_music_property_def__OVERRIDE__context_prop_spec__pr
    simple_music_property_def : OVERRIDE context_prop_spec property_path '=' scalar

LilyPondSyntacticalDefinition.p_simple_music_property_def__REVERT__context_prop_spec__embe
    simple_music_property_def : REVERT context_prop_spec embedded_scm

LilyPondSyntacticalDefinition.p_simple_music_property_def__SET__context_prop_spec__Chr61_
    simple_music_property_def : SET context_prop_spec '=' scalar

LilyPondSyntacticalDefinition.p_simple_music_property_def__UNSET__context_prop_spec(p)
    simple_music_property_def : UNSET context_prop_spec

LilyPondSyntacticalDefinition.p_simple_string__STRING(p)
    simple_string : STRING

LilyPondSyntacticalDefinition.p_simple_string__STRING_IDENTIFIER(p)
    simple_string : STRING_IDENTIFIER

LilyPondSyntacticalDefinition.p_simultaneous_music__DOUBLE_ANGLE_OPEN__music_list__DOUBLE_A
    simultaneous_music : DOUBLE_ANGLE_OPEN music_list DOUBLE_ANGLE_CLOSE

LilyPondSyntacticalDefinition.p_simultaneous_music__SIMULTANEOUS__braced_music_list(p)
    simultaneous_music : SIMULTANEOUS braced_music_list

LilyPondSyntacticalDefinition.p_start_symbol__lilypond(p)
    start_symbol : lilypond

LilyPondSyntacticalDefinition.p_steno_duration__DURATION_IDENTIFIER__dots(p)
    steno_duration : DURATION_IDENTIFIER dots

LilyPondSyntacticalDefinition.p_steno_duration__bare_unsigned__dots(p)
    steno_duration : bare_unsigned dots

LilyPondSyntacticalDefinition.p_steno_pitch__NOTENAME_PITCH(p)
    steno_pitch : NOTENAME_PITCH

LilyPondSyntacticalDefinition.p_steno_pitch__NOTENAME_PITCH__sub_quotes(p)
    steno_pitch : NOTENAME_PITCH sub_quotes

LilyPondSyntacticalDefinition.p_steno_pitch__NOTENAME_PITCH__sup_quotes(p)
    steno_pitch : NOTENAME_PITCH sup_quotes

LilyPondSyntacticalDefinition.p_steno_tonic_pitch__TONICNAME_PITCH(p)
    steno_tonic_pitch : TONICNAME_PITCH

LilyPondSyntacticalDefinition.p_steno_tonic_pitch__TONICNAME_PITCH__sub_quotes(p)
    steno_tonic_pitch : TONICNAME_PITCH sub_quotes

LilyPondSyntacticalDefinition.p_steno_tonic_pitch__TONICNAME_PITCH__sup_quotes(p)
    steno_tonic_pitch : TONICNAME_PITCH sup_quotes

LilyPondSyntacticalDefinition.p_string__STRING(p)
    string : STRING

LilyPondSyntacticalDefinition.p_string__STRING_IDENTIFIER(p)
    string : STRING_IDENTIFIER

LilyPondSyntacticalDefinition.p_string__string__Chr43__string(p)
    string : string '+' string

LilyPondSyntacticalDefinition.p_string_number_event__E_UNSIGNED(p)
    string_number_event : E_UNSIGNED

```

`LilyPondSyntacticalDefinition.p_sub_quotes__Chr44` (*p*)
`sub_quotes` : ‘,’

`LilyPondSyntacticalDefinition.p_sub_quotes__sub_quotes__Chr44` (*p*)
`sub_quotes` : `sub_quotes` ‘,’

`LilyPondSyntacticalDefinition.p_sup_quotes__Chr39` (*p*)
`sup_quotes` : ““

`LilyPondSyntacticalDefinition.p_sup_quotes__sup_quotes__Chr39` (*p*)
`sup_quotes` : `sup_quotes` ““

`LilyPondSyntacticalDefinition.p_tempo_event__TEMPO__scalar` (*p*)
`tempo_event` : TEMPO scalar

`LilyPondSyntacticalDefinition.p_tempo_event__TEMPO__scalar_closed__steno_duration__Chr61__t`
`tempo_event` : TEMPO scalar_closed steno_duration ‘=’ tempo_range

`LilyPondSyntacticalDefinition.p_tempo_event__TEMPO__steno_duration__Chr61__tempo_range` (*p*)
`tempo_event` : TEMPO steno_duration ‘=’ tempo_range

`LilyPondSyntacticalDefinition.p_tempo_range__bare_unsigned` (*p*)
`tempo_range` : bare_unsigned

`LilyPondSyntacticalDefinition.p_tempo_range__bare_unsigned__Chr126__bare_unsigned` (*p*)
`tempo_range` : bare_unsigned ‘~’ bare_unsigned

`LilyPondSyntacticalDefinition.p_toplevel_expression__composite_music` (*p*)
`toplevel_expression` : composite_music

`LilyPondSyntacticalDefinition.p_toplevel_expression__full_markup` (*p*)
`toplevel_expression` : full_markup

`LilyPondSyntacticalDefinition.p_toplevel_expression__full_markup_list` (*p*)
`toplevel_expression` : full_markup_list

`LilyPondSyntacticalDefinition.p_toplevel_expression__lilypond_header` (*p*)
`toplevel_expression` : lilypond_header

`LilyPondSyntacticalDefinition.p_toplevel_expression__output_def` (*p*)
`toplevel_expression` : output_def

`LilyPondSyntacticalDefinition.p_toplevel_expression__score_block` (*p*)
`toplevel_expression` : score_block

`LilyPondSyntacticalDefinition.p_tremolo_type__Chr58` (*p*)
`tremolo_type` : ‘.’

`LilyPondSyntacticalDefinition.p_tremolo_type__Chr58__bare_unsigned` (*p*)
`tremolo_type` : ‘.’ bare_unsigned

Special Methods

`LilyPondSyntacticalDefinition.__eq__` (*arg*)
 True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`LilyPondSyntacticalDefinition.__ge__` (*arg*)
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondSyntacticalDefinition.__gt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`LilyPondSyntacticalDefinition.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondSyntacticalDefinition.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`LilyPondSyntacticalDefinition.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

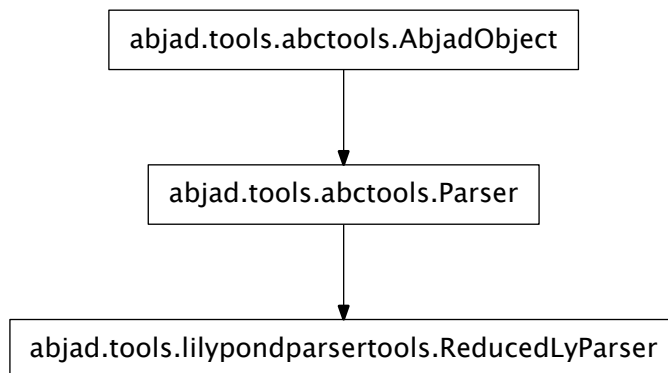
Inherited from `abctools.AbjadObject`

`LilyPondSyntacticalDefinition.__repr__()`
 Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`lilypondparsertools.ReducedLyParser`



class `lilypondparsertools.ReducedLyParser` (*debug=False*)
 Parses the “reduced-ly” syntax, a modified subset of LilyPond syntax:

```
>>> from abjad.tools import lilypondparsertools
>>> parser = lilypondparsertools.ReducedLyParser()
```

Understands LilyPond-like representation of notes, chords and rests:

```
>>> string = "c'4 r8. <b d' fs'>16"
>>> result = parser(string)
>>> f(result)
{
    c'4
    r8.
    <b d' fs'>16
}
```

Also parses bare duration as notes on middle-C, and negative bare durations as rests:

```
>>> string = '4 -8 16. -32'
>>> result = parser(string)
>>> f(result)
{
    c'4
    r8
    c'16.
    r32
}
```

Note that the leaf syntax is greedy, and therefore duration specifiers following pitch specifiers will be treated as part of the same expression. The following produces 2 leaves, rather than 3:

```
>>> string = "4 d' 4"
>>> result = parser(string)
>>> f(result)
{
    c'4
    d'4
}
```

Understands LilyPond-like default durations:

```
>>> string = "c'4 d' e' f'"
>>> result = parser(string)
>>> f(result)
{
    c'4
    d'4
    e'4
    f'4
}
```

Also understands various types of container specifications.

Can create arbitrarily nested tuplets:

```
>>> string = "2/3 { 4 4 3/5 { 8 8 8 } }"
>>> result = parser(string)
>>> f(result)
{
    \times 2/3 {
        c'4
    }
}
```

```

        c'4
        \fraction \times 3/5 {
            c'8
            c'8
            c'8
        }
    }
}

```

Can also create empty *FixedDurationContainers*:

```

>>> string = '{1/4} {2/4} {3/4} {4/4}'
>>> result = parser(string)
>>> for x in result: x
...
FixedDurationContainer(Duration(1, 4), [])
FixedDurationContainer(Duration(1, 2), [])
FixedDurationContainer(Duration(3, 4), [])
FixedDurationContainer(Duration(1, 1), [])

```

Can create measures too:

```

>>> string = '| 4/4 4 4 4 4 || 3/8 8 8 8 |'
>>> result = parser(string)
>>> for x in result: x
...
Measure(4/4, [c'4, c'4, c'4, c'4])
Measure(3/8, [c'8, c'8, c'8])

```

Finally, understands ties, slurs and beams:

```

>>> string = 'c16 [ ( d ~ d ) f ]'
>>> result = parser(string)
>>> f(result)
{
    c16 [ (
        d16 ~
        d16 )
    f16 ]
}

```

Return *ReducedLyParser* instance.

Read-only Properties

`ReducedLyParser.debug`

True if the parser runs in debugging mode.

Inherited from `abctools.Parser`

`ReducedLyParser.lexer`

The parser's PLY Lexer instance.

Inherited from `abctools.Parser`

`ReducedLyParser.lexer_rules_object`

`ReducedLyParser.logger`

The parser's Logger instance.

Inherited from `abctools.Parser`

`ReducedLyParser.logger_path`

The output path for the parser's logfile.

Inherited from `abctools.Parser`

`ReducedLyParser.output_path`

The output path for files associated with the parser.

Inherited from `abctools.Parser`

`ReducedLyParser.parser`

The parser's PLY LRPParser instance.

Inherited from `abctools.Parser`

`ReducedLyParser.parser_rules_object`

`ReducedLyParser.pickle_path`

The output path for the parser's pickled parsing tables.

Inherited from `abctools.Parser`

`ReducedLyParser.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`ReducedLyParser.p_apostrophes__APOSTROPHE` (*p*)

apostrophes : APOSTROPHE

`ReducedLyParser.p_apostrophes__apostrophes__APOSTROPHE` (*p*)

apostrophes : apostrophes APOSTROPHE

`ReducedLyParser.p_beam__BRACKET_L` (*p*)

beam : BRACKET_L

`ReducedLyParser.p_beam__BRACKET_R` (*p*)

beam : BRACKET_R

`ReducedLyParser.p_chord_body__chord_pitches` (*p*)

chord_body : chord_pitches

`ReducedLyParser.p_chord_body__chord_pitches__positive_leaf_duration` (*p*)

chord_body : chord_pitches positive_leaf_duration

`ReducedLyParser.p_chord_pitches__CARAT_L__pitches__CARAT_R` (*p*)

chord_pitches : CARAT_L pitches CARAT_R

`ReducedLyParser.p_commas__COMMA` (*p*)

commas : COMMA

`ReducedLyParser.p_commas__commas__commas` (*p*)

commas : commas COMMA

`ReducedLyParser.p_component__container` (*p*)

component : container

`ReducedLyParser.p_component__fixed_duration_container` (*p*)

component : fixed_duration_container

`ReducedLyParser.p_component__leaf` (*p*)

component : leaf

```

ReducedLyParser.p_component__tuple (p)
    component : tuple

ReducedLyParser.p_component_list__EMPTY (p)
    component_list :

ReducedLyParser.p_component_list__component_list__component (p)
    component_list : component_list component

ReducedLyParser.p_container__BRACE_L__component_list__BRACE_R (p)
    container : BRACE_L component_list BRACE_R

ReducedLyParser.p_dots__EMPTY (p)
    dots :

ReducedLyParser.p_dots__dots__DOT (p)
    dots : dots DOT

ReducedLyParser.p_error (p)

ReducedLyParser.p_fixed_duration_container__BRACE_L__FRACTION__BRACE_R (p)
    fixed_duration_container : BRACE_L FRACTION BRACE_R

ReducedLyParser.p_leaf__leaf_body__post_events (p)
    leaf : leaf_body post_events

ReducedLyParser.p_leaf_body__chord_body (p)
    leaf_body : chord_body

ReducedLyParser.p_leaf_body__note_body (p)
    leaf_body : note_body

ReducedLyParser.p_leaf_body__rest_body (p)
    leaf_body : rest_body

ReducedLyParser.p_measure__PIPE__FRACTION__component_list__PIPE (p)
    measure : PIPE FRACTION component_list PIPE

ReducedLyParser.p_negative_leaf_duration__INTEGER_N__dots (p)
    negative_leaf_duration : INTEGER_N dots

ReducedLyParser.p_note_body__pitch (p)
    note_body : pitch

ReducedLyParser.p_note_body__pitch__positive_leaf_duration (p)
    note_body : pitch positive_leaf_duration

ReducedLyParser.p_note_body__positive_leaf_duration (p)
    note_body : positive_leaf_duration

ReducedLyParser.p_pitch__PITCHNAME (p)
    pitch : PITCHNAME

ReducedLyParser.p_pitch__PITCHNAME__apostrophes (p)
    pitch : PITCHNAME apostrophes

ReducedLyParser.p_pitch__PITCHNAME__commas (p)
    pitch : PITCHNAME commas

ReducedLyParser.p_pitches__pitch (p)
    pitches : pitch

ReducedLyParser.p_pitches__pitches__pitch (p)
    pitches : pitches pitch

```

```

ReducedLyParser.p_positive_leaf_duration__INTEGER_P__dots (p)
    positive_leaf_duration : INTEGER_P dots

ReducedLyParser.p_post_event__beam (p)
    post_event : beam

ReducedLyParser.p_post_event__slur (p)
    post_event : slur

ReducedLyParser.p_post_event__tie (p)
    post_event : tie

ReducedLyParser.p_post_events__EMPTY (p)
    post_events :

ReducedLyParser.p_post_events__post_events__post_event (p)
    post_events : post_events post_event

ReducedLyParser.p_rest_body__RESTNAME (p)
    rest_body : RESTNAME

ReducedLyParser.p_rest_body__RESTNAME__positive_leaf_duration (p)
    rest_body : RESTNAME positive_leaf_duration

ReducedLyParser.p_rest_body__negative_leaf_duration (p)
    rest_body : negative_leaf_duration

ReducedLyParser.p_slur__PAREN_L (p)
    slur : PAREN_L

ReducedLyParser.p_slur__PAREN_R (p)
    slur : PAREN_R

ReducedLyParser.p_start__EMPTY (p)
    start :

ReducedLyParser.p_start__start__component (p)
    start : start component

ReducedLyParser.p_start__start__measure (p)
    start : start measure

ReducedLyParser.p_tie__TILDE (p)
    tie : TILDE

ReducedLyParser.p_tuplet__FRACTION__container (p)
    tuplet : FRACTION container

ReducedLyParser.t_FRACTION (t)
    ([1-9]d*/[1-9]d*)

ReducedLyParser.t_INTEGER_N (t)
    (-[1-9]d*)

ReducedLyParser.t_INTEGER_P (t)
    ([1-9]d*)

ReducedLyParser.t_PITCHNAME (t)
    [a-g](fflsslflslqtqltqlslqlf)?

ReducedLyParser.t_error (t)

ReducedLyParser.t_newline (t)
    n+

```


`ReducedLyParser.tokenize(input_string)`

Tokenize *input string* and print results.

Inherited from `abctools.Parser`

Special Methods

`ReducedLyParser.__call__(input_string)`

Parse *input_string* and return result.

Inherited from `abctools.Parser`

`ReducedLyParser.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ReducedLyParser.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReducedLyParser.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ReducedLyParser.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReducedLyParser.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ReducedLyParser.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

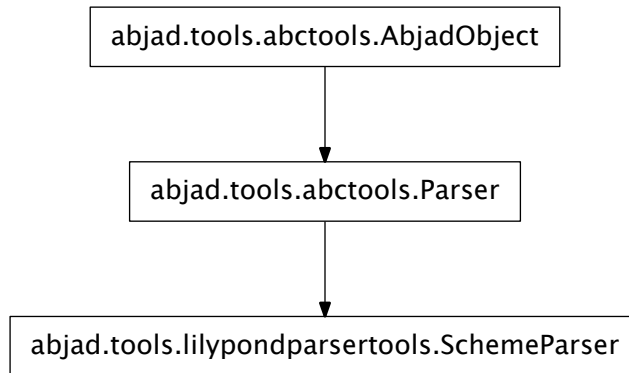
`ReducedLyParser.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`lilypondparsertools.SchemeParser`



class `lilypondparsertools.SchemeParser` (*debug=False*)
SchemeParser mimics how LilyPond’s embedded Scheme parser behaves.
 It parses a single Scheme expression and then stops, by raising a *SchemeParserFinishedException*.
 The parsed expression and its exact length in characters are cached on the *SchemeParser* instance.
 It is intended to be used only in conjunction with *LilyPondParser*.
 Returns *SchemeParser* instance.

Read-only Properties

`SchemeParser.debug`
 True if the parser runs in debugging mode.

Inherited from `abctools.Parser`

`SchemeParser.lexer`
 The parser’s PLY Lexer instance.

Inherited from `abctools.Parser`

`SchemeParser.lexer_rules_object`

`SchemeParser.logger`
 The parser’s Logger instance.

Inherited from `abctools.Parser`

`SchemeParser.logger_path`
 The output path for the parser’s logfile.

Inherited from `abctools.Parser`

`SchemeParser.output_path`
 The output path for files associated with the parser.

Inherited from `abctools.Parser`

`SchemeParser.parser`

The parser's PLY LRParser instance.

Inherited from `abctools.Parser`

`SchemeParser.parser_rules_object`

`SchemeParser.pickle_path`

The output path for the parser's pickled parsing tables.

Inherited from `abctools.Parser`

`SchemeParser.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`SchemeParser.p_boolean__BOOLEAN(p)`

boolean : BOOLEAN

`SchemeParser.p_constant__boolean(p)`

constant : boolean

`SchemeParser.p_constant__number(p)`

constant : number

`SchemeParser.p_constant__string(p)`

constant : string

`SchemeParser.p_data__EMPTY(p)`

data :

`SchemeParser.p_data__data__datum(p)`

data : data datum

`SchemeParser.p_datum__constant(p)`

datum : constant

`SchemeParser.p_datum__list(p)`

datum : list

`SchemeParser.p_datum__symbol(p)`

datum : symbol

`SchemeParser.p_datum__vector(p)`

datum : vector

`SchemeParser.p_error(p)`

`SchemeParser.p_expression__QUOTE__datum(p)`

expression : QUOTE datum

`SchemeParser.p_expression__constant(p)`

expression : constant

`SchemeParser.p_expression__variable(p)`

expression : variable

`SchemeParser.p_form__expression(p)`

form : expression

```

SchemeParser.p_forms__EMPTY(p)
    forms :
SchemeParser.p_forms__forms__form(p)
    forms : forms form
SchemeParser.p_list__L_PAREN__data__R_PAREN(p)
    list : L_PAREN data R_PAREN
SchemeParser.p_list__L_PAREN__data__datum__PERIOD__datum__R_PAREN(p)
    list : L_PAREN data datum PERIOD datum R_PAREN
SchemeParser.p_number__DECIMAL(p)
    number : DECIMAL
SchemeParser.p_number__INTEGER(p)
    number : INTEGER
SchemeParser.p_program__forms(p)
    program : forms
SchemeParser.p_string__STRING(p)
    string : STRING
SchemeParser.p_symbol__IDENTIFIER(p)
    symbol : IDENTIFIER
SchemeParser.p_variable__IDENTIFIER(p)
    variable : IDENTIFIER
SchemeParser.p_vector__HASH__L_PAREN__data__R_PAREN(p)
    vector : HASH L_PAREN data R_PAREN
SchemeParser.t_BOOLEAN(t)
    #(T|F|t|f)
SchemeParser.t_DECIMAL(t)
    (((-?[0-9]+).[0-9]*)|(-?[0-9]+))
SchemeParser.t_HASH(t)
    #
SchemeParser.t_IDENTIFIER(t)
    ([A-Za-z!$%&*/<>?~_^:=][A-Za-z0-9!$%&*/<>?~_^:=]*|[-+]|...)
SchemeParser.t_INTEGER(t)
    (-?[0-9]+)
SchemeParser.t_L_PAREN(t)
    (
SchemeParser.t_R_PAREN(t)
    )
SchemeParser.t_anything(t)
    .
SchemeParser.t_error(t)
SchemeParser.t_newline(t)
    n+
SchemeParser.t_quote(t)
    “

```

`SchemeParser.t_quote_440(t)`
`\[nt\"]`

`SchemeParser.t_quote_443(t)`
`[^\"]+`

`SchemeParser.t_quote_446(t)`
`“`

`SchemeParser.t_quote_456(t)`
`.`

`SchemeParser.t_quote_error(t)`

`SchemeParser.t_whitespace(t)`
`[\t]+`

`SchemeParser.tokenize(input_string)`
 Tokenize *input string* and print results.
 Inherited from `abctools.Parser`

Special Methods

`SchemeParser.__call__(input_string)`
 Parse *input_string* and return result.
 Inherited from `abctools.Parser`

`SchemeParser.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.
 Inherited from `abctools.AbjadObject`

`SchemeParser.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`SchemeParser.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception
 Inherited from `abctools.AbjadObject`

`SchemeParser.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`SchemeParser.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.
 Inherited from `abctools.AbjadObject`

`SchemeParser.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

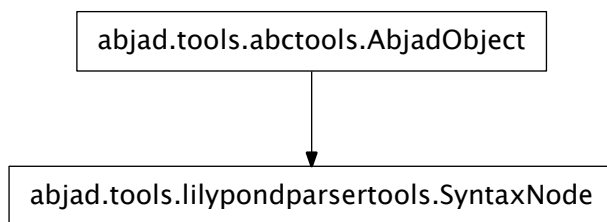
`SchemeParser.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`lilypondparsertools.SyntaxNode`



class `lilypondparsertools.SyntaxNode`(*type*, *value*=[])

A node in an abstract syntax tree (AST).

Not composer-safe.

Used internally by `LilyPondParser`.

Read-only Properties

`SyntaxNode.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Special Methods

`SyntaxNode.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SyntaxNode.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SyntaxNode.__getitem__(item)`

`SyntaxNode.__gt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`SyntaxNode.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SyntaxNode.__len__()`

`SyntaxNode.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`SyntaxNode.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`SyntaxNode.__repr__()`

`SyntaxNode.__str__()`

functions

`lilypondparsertools.lilypond_enharmonic_transpose`

`lilypondparsertools.lilypond_enharmonic_transpose(pitch_a, pitch_b, pitch_c)`
 Transpose *pitch_c* by the distance between *pitch_b* and *pitch_a*.

This function was reverse-engineered from LilyPond's source code.

Return `NamedChromaticPitch`.

`lilypondparsertools.parse_reduced_ly_syntax`

`lilypondparsertools.parse_reduced_ly_syntax(string)`
 Parse the reduced LilyPond rhythmic syntax:

```
>>> from abjad.tools import lilypondparsertools

>>> string = '4 -4. 8.. 5/3 { } 4'
>>> result = lilypondparsertools.parse_reduced_ly_syntax(string)

>>> for x in result:
...     x
...
Note("c'4")
Rest('r4.')
Note("c'8..")
```

```
Tuplet(5/3, [])  
Note("c'4")
```

Return list.

offsettools

functions

offsettools.update_offset_values_of_component

`offsettools.update_offset_values_of_component(component)`
New in version 2.9. Update prolated offset values of *component*.

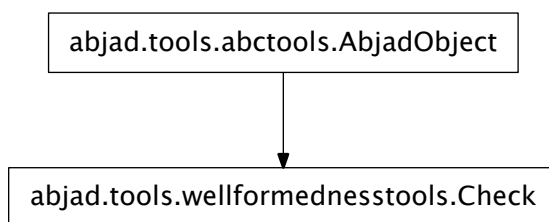
offsettools.update_offset_values_of_component_in_seconds

`offsettools.update_offset_values_of_component_in_seconds(component)`
New in version 2.9. Update offset values of *component* in seconds.

wellformednesstools

abstract classes

wellformednesstools.Check



class `wellformednesstools.Check`

Read-only Properties

`Check.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`Check.check` (*expr*)

`Check.report` (*expr*)

`Check.violators` (*expr*)

Special Methods

`Check.__eq__` (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Check.__ge__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Check.__gt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`Check.__le__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Check.__lt__` (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`Check.__ne__` (*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`Check.__repr__` ()

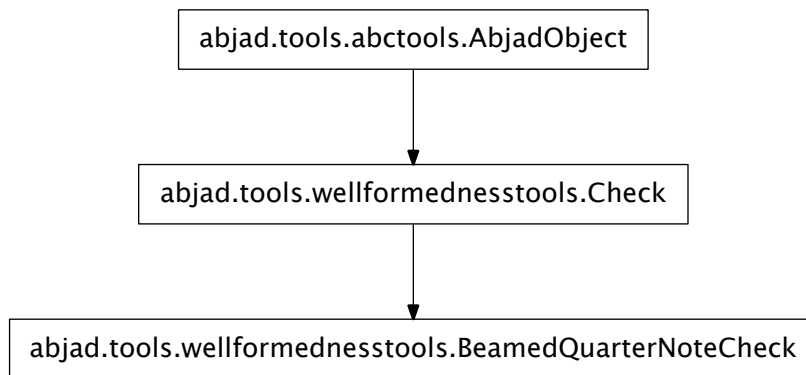
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

concrete classes

wellformednesstools.BeamedQuarterNoteCheck



class `wellformednesstools.BeamedQuarterNoteCheck`

Read-only Properties

`BeamedQuarterNoteCheck.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`BeamedQuarterNoteCheck.check(expr)`

Inherited from `wellformednesstools.Check`

`BeamedQuarterNoteCheck.report(expr)`

Inherited from `wellformednesstools.Check`

`BeamedQuarterNoteCheck.violators(expr)`

Inherited from `wellformednesstools.Check`

Special Methods

`BeamedQuarterNoteCheck.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

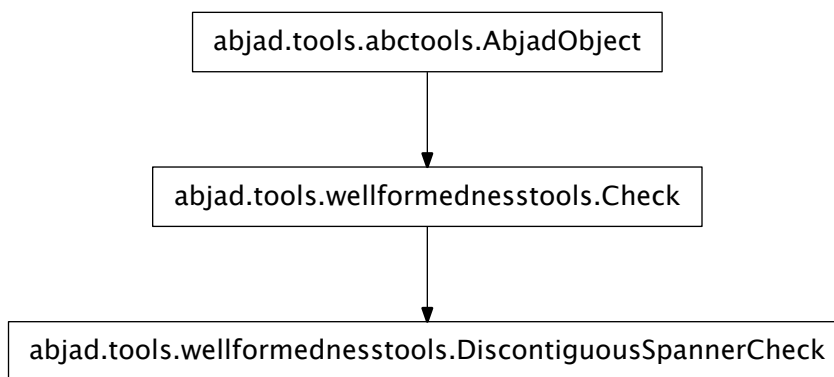
`BeamedQuarterNoteCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

wellformednesstools.DiscontiguousSpannerCheck



class wellformednesstools.DiscontiguousSpannerCheck

There are now two different types of spanner. Most spanners demand that spanner components be thread-contiguous. But a few special spanners (like Tempo) do not make such a demand. The check here consults the experimental `_contiguity_constraint`.

Read-only Properties

`DiscontiguousSpannerCheck.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`DiscontiguousSpannerCheck.check(expr)`
Inherited from `wellformednesstools.Check`

`DiscontiguousSpannerCheck.report(expr)`
Inherited from `wellformednesstools.Check`

`DiscontiguousSpannerCheck.violators(expr)`
Inherited from `wellformednesstools.Check`

Special Methods

`DiscontiguousSpannerCheck.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

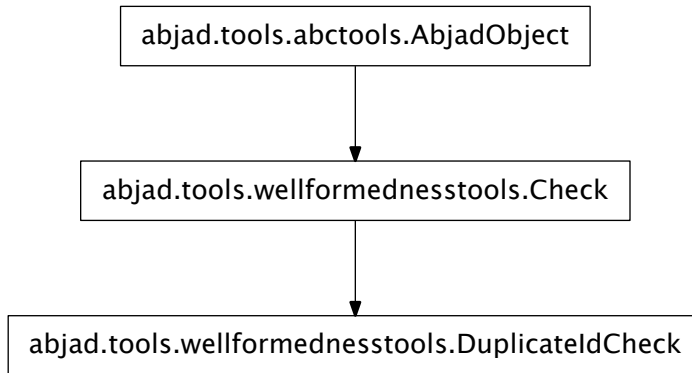
Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`wellformednesstools.DuplicateIdCheck`



class `wellformednesstools.DuplicateIdCheck`

Read-only Properties

`DuplicateIdCheck.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`DuplicateIdCheck.check(expr)`

Inherited from `wellformednesstools.Check`

`DuplicateIdCheck.report(expr)`

Inherited from `wellformednesstools.Check`

`DuplicateIdCheck.violators(expr)`

Inherited from `wellformednesstools.Check`

Special Methods

`DuplicateIdCheck.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

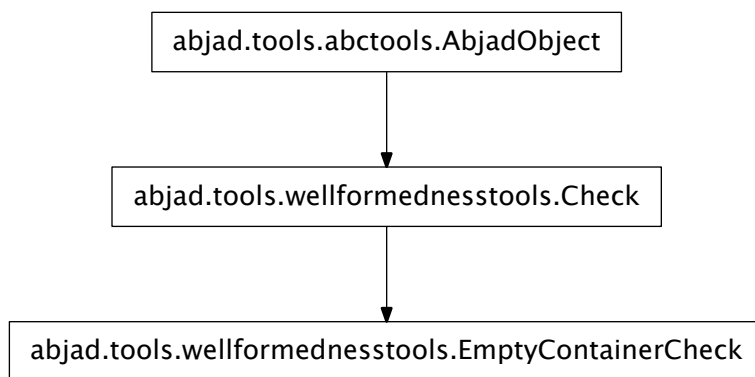
`DuplicateIdCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

wellformednesstools.EmptyContainerCheck



```
class wellformednesstools.EmptyContainerCheck
```

Read-only Properties

`EmptyContainerCheck.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`EmptyContainerCheck.check(expr)`

Inherited from `wellformednesstools.Check`

`EmptyContainerCheck.report(expr)`

Inherited from `wellformednesstools.Check`

`EmptyContainerCheck.violators(expr)`

Inherited from `wellformednesstools.Check`

Special Methods

`EmptyContainerCheck.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`EmptyContainerCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`EmptyContainerCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`EmptyContainerCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`EmptyContainerCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`EmptyContainerCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

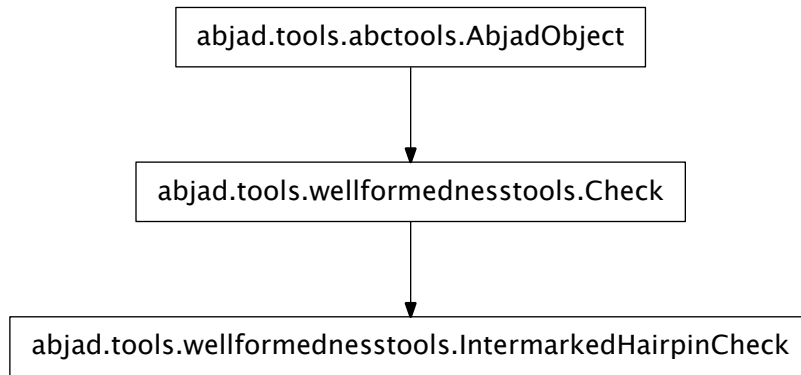
`EmptyContainerCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`wellformednesstools.IntermarkedHairpinCheck`



class `wellformednesstools.IntermarkedHairpinCheck`
 Are there any dynamic marks in the middle of a hairpin?

Read-only Properties

`IntermarkedHairpinCheck.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`IntermarkedHairpinCheck.check(expr)`
 Inherited from `wellformednesstools.Check`

`IntermarkedHairpinCheck.report(expr)`
 Inherited from `wellformednesstools.Check`

`IntermarkedHairpinCheck.violators(expr)`
 Inherited from `wellformednesstools.Check`

Special Methods

`IntermarkedHairpinCheck.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__ge__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

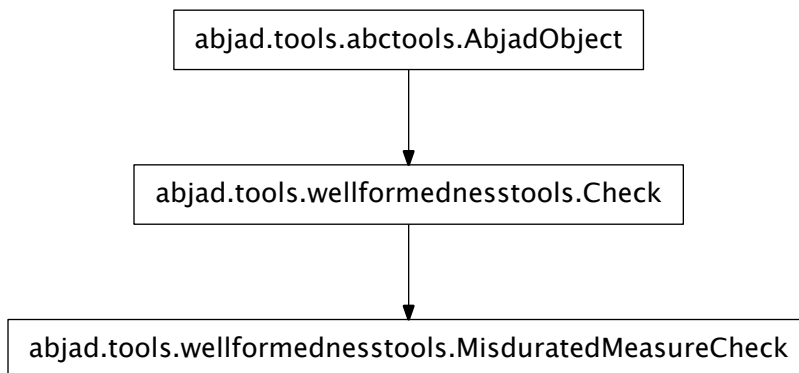
`IntermarkedHairpinCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

wellformednesstools.MisduratedMeasureCheck



class `wellformednesstools.MisduratedMeasureCheck`

Does the (pre)rolated duration of the measure match its meter?

Read-only Properties

`MisdatedMeasureCheck.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`MisdatedMeasureCheck.check(expr)`

Inherited from `wellformednesstools.Check`

`MisdatedMeasureCheck.report(expr)`

Inherited from `wellformednesstools.Check`

`MisdatedMeasureCheck.violators(expr)`

Inherited from `wellformednesstools.Check`

Special Methods

`MisdatedMeasureCheck.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MisdatedMeasureCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MisdatedMeasureCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MisdatedMeasureCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MisdatedMeasureCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MisdatedMeasureCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

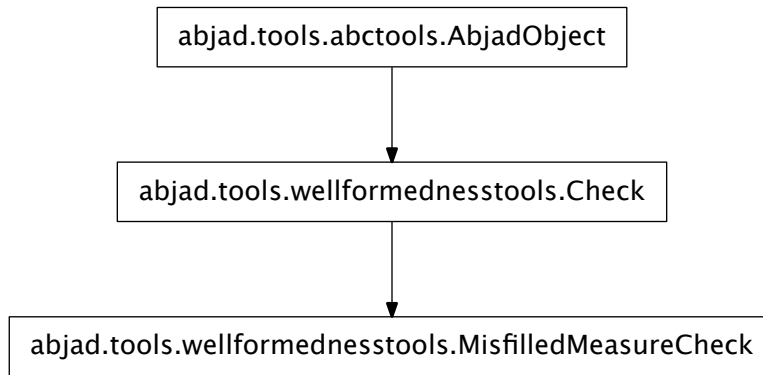
`MisdatedMeasureCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`wellformednesstools.MisfilledMeasureCheck`



class `wellformednesstools.MisfilledMeasureCheck`

Check that time signature duration equals measure contents duration for every measure.

Read-only Properties

`MisfilledMeasureCheck.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`MisfilledMeasureCheck.check(expr)`

Inherited from `wellformednesstools.Check`

`MisfilledMeasureCheck.report(expr)`

Inherited from `wellformednesstools.Check`

`MisfilledMeasureCheck.violators(expr)`

Inherited from `wellformednesstools.Check`

Special Methods

`MisfilledMeasureCheck.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

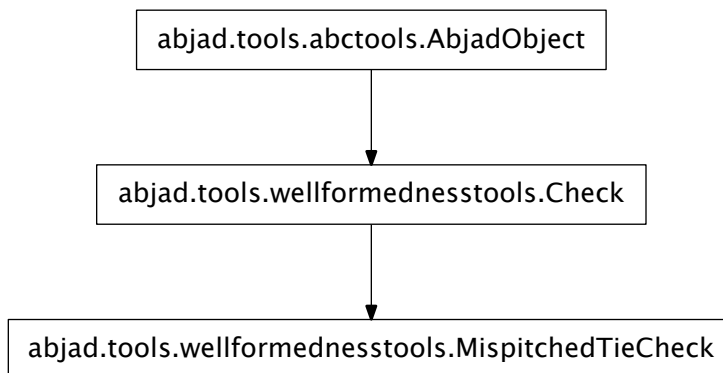
`MisfilledMeasureCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

wellformednesstools.MispitchedTieCheck



```
class wellformednesstools.MispitchedTieCheck
```

Read-only Properties

`MispitchedTieCheck.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`MispitchedTieCheck.check(expr)`

Inherited from `wellformednesstools.Check`

`MispitchedTieCheck.report(expr)`

Inherited from `wellformednesstools.Check`

`MispitchedTieCheck.violators(expr)`

Inherited from `wellformednesstools.Check`

Special Methods

`MispitchedTieCheck.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MispitchedTieCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MispitchedTieCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MispitchedTieCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MispitchedTieCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MispitchedTieCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

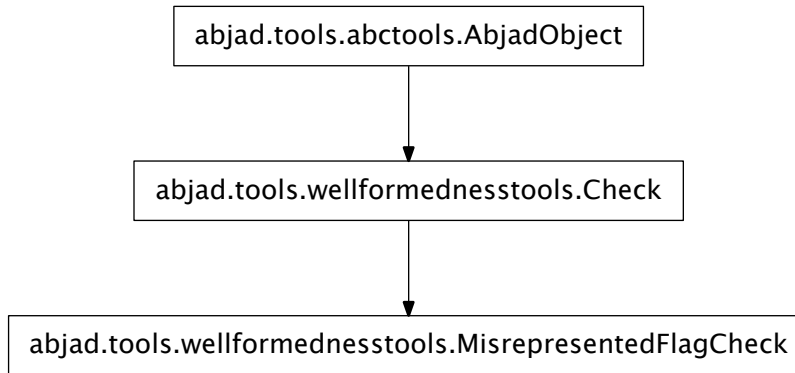
`MispitchedTieCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`wellformednesstools.MisrepresentedFlagCheck`



class `wellformednesstools.MisrepresentedFlagCheck`

Read-only Properties

`MisrepresentedFlagCheck.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`MisrepresentedFlagCheck.check(expr)`

Inherited from `wellformednesstools.Check`

`MisrepresentedFlagCheck.report(expr)`

Inherited from `wellformednesstools.Check`

`MisrepresentedFlagCheck.violators(expr)`

Inherited from `wellformednesstools.Check`

Special Methods

`MisrepresentedFlagCheck.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MisrepresentedFlagCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MisrepresentedFlagCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MisrepresentedFlagCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MisrepresentedFlagCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MisrepresentedFlagCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

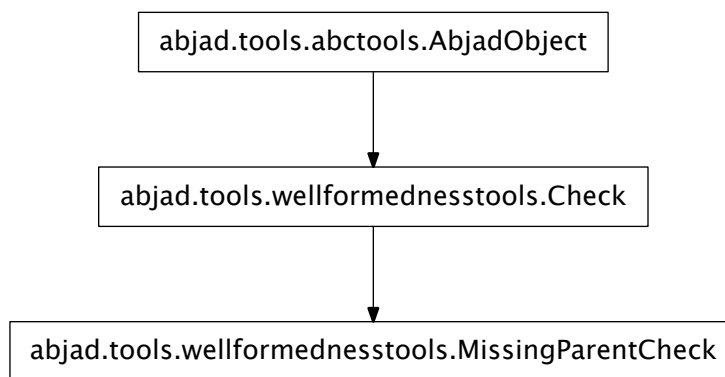
`MisrepresentedFlagCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`wellformednesstools.MissingParentCheck`



class `wellformednesstools.MissingParentCheck`

Each node except the root needs a parent.

Read-only Properties

`MissingParentCheck.storage_format`

Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`MissingParentCheck.check(expr)`

Inherited from `wellformednesstools.Check`

`MissingParentCheck.report(expr)`

Inherited from `wellformednesstools.Check`

`MissingParentCheck.violators(expr)`

Inherited from `wellformednesstools.Check`

Special Methods

`MissingParentCheck.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`MissingParentCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MissingParentCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`MissingParentCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MissingParentCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`MissingParentCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

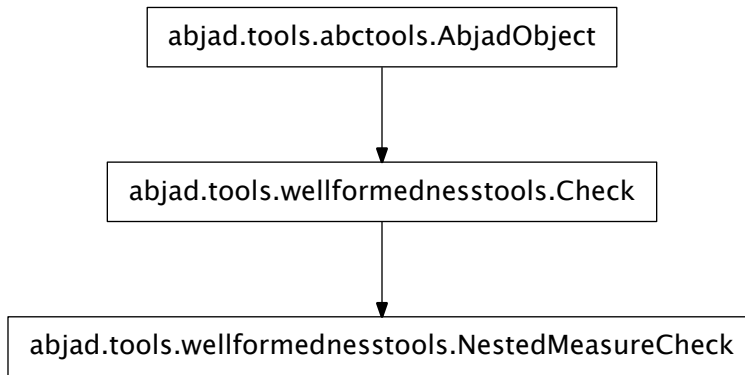
`MissingParentCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`wellformednesstools.NestedMeasureCheck`



class `wellformednesstools.NestedMeasureCheck`
 Do we have any nested measures?

Read-only Properties

`NestedMeasureCheck.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`NestedMeasureCheck.check(expr)`
 Inherited from `wellformednesstools.Check`

`NestedMeasureCheck.report(expr)`
 Inherited from `wellformednesstools.Check`

`NestedMeasureCheck.violators(expr)`
 Inherited from `wellformednesstools.Check`

Special Methods

`NestedMeasureCheck.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__ge__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

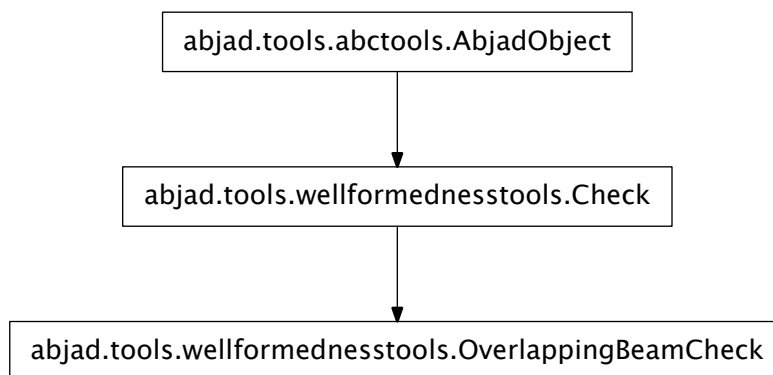
`NestedMeasureCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

wellformednesstools.OverlappingBeamCheck



class `wellformednesstools.OverlappingBeamCheck`

Beams must not overlap.

Read-only Properties

`OverlappingBeamCheck.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`OverlappingBeamCheck.check(expr)`
Inherited from `wellformednesstools.Check`

`OverlappingBeamCheck.report(expr)`
Inherited from `wellformednesstools.Check`

`OverlappingBeamCheck.violators(expr)`
Inherited from `wellformednesstools.Check`

Special Methods

`OverlappingBeamCheck.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

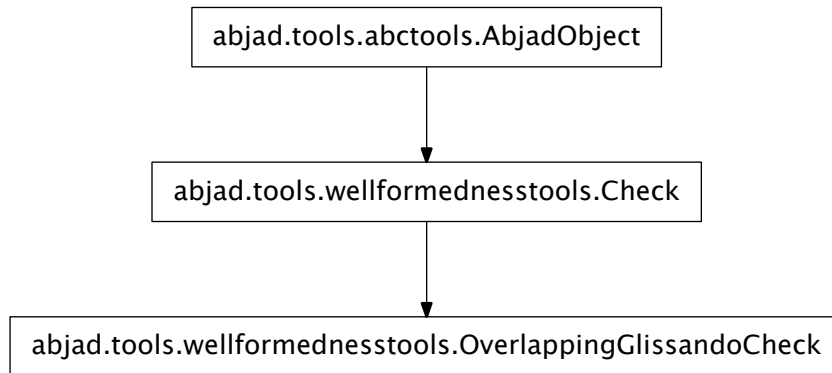
Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`wellformednesstools.OverlappingGlissandoCheck`



class `wellformednesstools.OverlappingGlissandoCheck`
Glissandi must not overlap. Dove-tailed glissandi are OK.

Read-only Properties

`OverlappingGlissandoCheck.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`OverlappingGlissandoCheck.check(expr)`
Inherited from `wellformednesstools.Check`

`OverlappingGlissandoCheck.report(expr)`
Inherited from `wellformednesstools.Check`

`OverlappingGlissandoCheck.violators(expr)`
Inherited from `wellformednesstools.Check`

Special Methods

`OverlappingGlissandoCheck.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__gt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.

Return boolean.

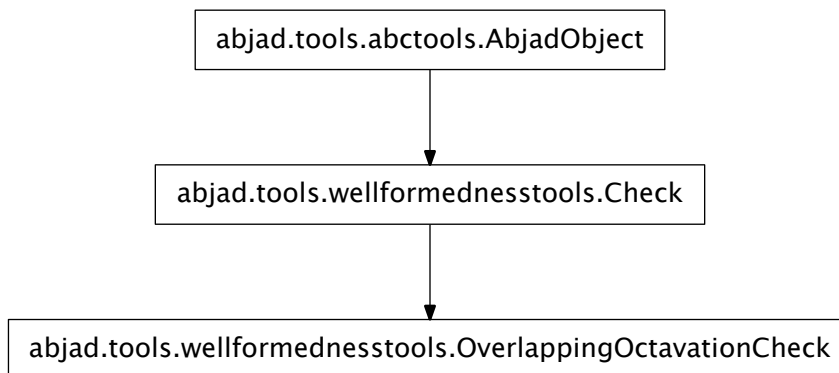
Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__repr__()`
 Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

wellformednesstools.OverlappingOctavationCheck



class `wellformednesstools.OverlappingOctavationCheck`
 Octavation spanners must not overlap.

Read-only Properties

`OverlappingOctavationCheck.storage_format`
Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`OverlappingOctavationCheck.check(expr)`
Inherited from `wellformednesstools.Check`

`OverlappingOctavationCheck.report(expr)`
Inherited from `wellformednesstools.Check`

`OverlappingOctavationCheck.violators(expr)`
Inherited from `wellformednesstools.Check`

Special Methods

`OverlappingOctavationCheck.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

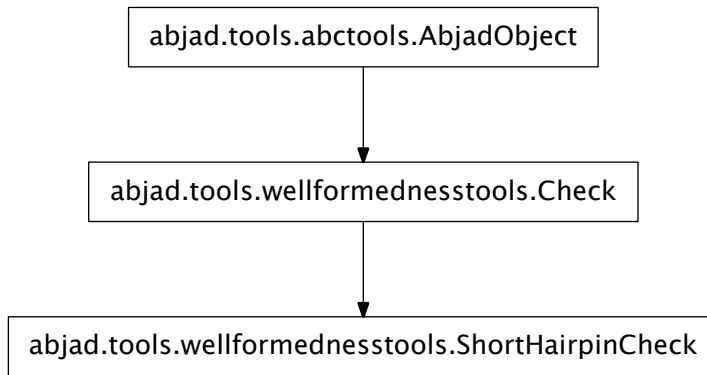
Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

`wellformednesstools.ShortHairpinCheck`



class `wellformednesstools.ShortHairpinCheck`
 Hairpins must span at least two leaves.

Read-only Properties

`ShortHairpinCheck.storage_format`
 Storage format of Abjad object.

Return string.

Inherited from `abctools.AbjadObject`

Methods

`ShortHairpinCheck.check(expr)`
 Inherited from `wellformednesstools.Check`

`ShortHairpinCheck.report(expr)`
 Inherited from `wellformednesstools.Check`

`ShortHairpinCheck.violators(expr)`
 Inherited from `wellformednesstools.Check`

Special Methods

`ShortHairpinCheck.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__ge__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Inherited from `abctools.AbjadObject`

functions

`wellformednesstools.list_checks`

`wellformednesstools.list_checks()`

New in version 2.8. List checks:

```
>>> from abjad.tools import wellformednesstools

>>> for check in wellformednesstools.list_checks(): check
...
BeamedQuarterNoteCheck()
DiscontiguousSpannerCheck()
DuplicateIdCheck()
EmptyContainerCheck()
IntermarkedHairpinCheck()
MisduratedMeasureCheck()
MisfilledMeasureCheck()
MispitchedTieCheck()
MisrepresentedFlagCheck()
MissingParentCheck()
NestedMeasureCheck()
OverlappingBeamCheck()
OverlappingGlissandoCheck()
```



```
OverlappingOctavationCheck()  
ShortHairpinCheck()
```

Return list of checks.

BIBLIOGRAPHY

- [Adan2006] Víctor Adán. Music <-> Geometry <-> Meta-Music. Draft February 12, 2006.
- [AgonAssayagBresson2006] Carlos Agon, Gérard Assayag, Jean Bresson. The OM Composer's Book 1. Éditions Delatour, Paris. 2006.
- [AgonHaddadAssayag2002] Carlos Agon, Karim Haddad & Gerard Assayag. Représentation et rendu de structures rythmiques. Journées d'Informatique Musicale, 9th ed., Marseille, 29 - 31 May 2002.
- [Alegant1993] Brian Alegant. The seventy-seven partitions of the aggregate: Analytical and theoretical implications. Doctoral Dissertation. The University of Rochester, Eastman School of Music. 1993.
- [Ariza2005] Christopher Ariza. An Open Design for Computer-Aided Algorithmic Music Composition: athenaCL. Dissertation.com, Boca Raton. 2005.
- [BacaAdan2007] Trevor Bača & Víctor Adán. Cuepatlahto and Lascaux: two approaches to the formalized control of musical score. Draft June 7, 2007.
- [BressonAgonAssayag2008] Jean Bresson, Carlos Agon, Gérard Assayag. The OM Composer's Book 2. Éditions Delatour, Paris. 2008.
- [Carter2002] Eliot Carter. Harmony Book. Nicholas Hopkins and John F. Link, eds. Carl Fischer, New York. 2002.
- [Haddad] Karim Haddad. Le Temps comme Territoire: pour une géographie temporelle.
- [Kampela1998] Arthur Kampela. Uma Faca Só Lâmina. Doctoral Dissertation. Columbia University, NY, NY. 1998.
- [Malt2008] Mikhail Malt. Some Considerations on Brian Ferneyhough's Musical Language Through His Use of CAC – Part I: Time and Rhythmic Structures. In Bresson, Agon and Assayag (2008).
- [Morris1987] Robert Morris. Composition with Pitch-Classes. Yale University Press, New Haven. 1987.
- [Nauert1997] Paul Nauert. Timespan Formation in Nonmetric, Posttonal Music. Doctoral Dissertation. Columbia University, NY, NY. 1997.
- [NienhuysNieuwenhuizen2003] Han-Wen Nienhuys & Jan Nieuwenhuizen. Lilypond: A system for automated music engraving. Proceedings of the XIV Colloquium on Musical Informatics. Firenze, Italy. May 8 - 10, 2003.
- [Ross1987] Ted Ross. Teach Yourself The Art of Music Engraving and Processing. Hansen House, Miami Beach. 1987.
- [Selfridge-Field1997] Eleanor Selfridge-Field, ed. Beyond MIDI: The Handbook of Musical Codes. The MIT Press. Cambridge, Massachusetts. 1997.
- [Valle] Andrea Valle. GeoGraphy: Notazione musicale e composizione algoritmica. Centro Interdipartimentale di Ricerca sulla Multimedialità e l'Audiovisivo. Università degli Studi di Torino.
- [WulfsonBarrettWinter] Harris Wulfson, G. Douglas Barrett & Michael Winter. Automatic Notation Generators.

INDEX

Symbols

- `__abs__()` (abjad.tools.durationtools.Duration.Duration.Duration method), 447
- `__abs__()` (abjad.tools.durationtools.Offset.Offset.Offset method), 451
- `__abs__()` (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction method), 949
- `__abs__()` (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject.ChromaticIntervalClassObject method), 1064
- `__abs__()` (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.ChromaticIntervalObject method), 1066
- `__abs__()` (abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject.ChromaticPitchObject method), 1069
- `__abs__()` (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject.CounterpointIntervalClassObject method), 1071
- `__abs__()` (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.CounterpointIntervalObject method), 1072
- `__abs__()` (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject.DiatonicIntervalClassObject method), 1075
- `__abs__()` (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.DiatonicIntervalObject method), 1077
- `__abs__()` (abjad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClassObject.DiatonicPitchClassObject method), 1079
- `__abs__()` (abjad.tools.pitchtools.DiatonicPitchObject.DiatonicPitchObject.DiatonicPitchObject method), 1081
- `__abs__()` (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval method), 1131
- `__abs__()` (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 1133
- `__abs__()` (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 1143
- `__abs__()` (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1144
- `__abs__()` (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval method), 1146
- `__abs__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1147
- `__abs__()` (abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject.HarmonicIntervalClassObject method), 1083
- `__abs__()` (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.HarmonicIntervalObject method), 1084
- `__abs__()` (abjad.tools.pitchtools.IntervalObject.IntervalObject.IntervalObject method), 1092
- `__abs__()` (abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass.IntervalObjectClass method), 1093
- `__abs__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1158
- `__abs__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass method), 1167
- `__abs__()` (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval method), 1175
- `__abs__()` (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1177
- `__abs__()` (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval method), 1190
- `__abs__()` (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass method), 1191
- `__abs__()` (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1194
- `__abs__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass method), 1196
- `__abs__()` (abjad.tools.pitchtools.MelodicIntervalClassObject.MelodicIntervalClassObject.MelodicIntervalClassObject method), 1199
- `__abs__()` (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject.MelodicIntervalObject method), 1192
- `__abs__()` (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch method), 1206
- `__abs__()` (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1208
- `__abs__()` (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch method), 1226
- `__abs__()` (abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass.NamedDiatonicPitchClass method), 1228
- `__abs__()` (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch.NumberedChromaticPitch method), 1231
- `__abs__()` (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1233
- `__abs__()` (abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch.NumberedDiatonicPitch method), 1247
- `__abs__()` (abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass method), 1249
- `__abs__()` (abjad.tools.pitchtools.NumberedPitchObject.NumberedPitchObject.NumberedPitchObject method), 1108

`__abs__()` (abjad.tools.pitchtools.PitchObject.PitchObject.PitchObject) (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject) method), 1123
`__add__()` (abjad.tools.containertools.Cluster.Cluster.Cluster) (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval) method), 340
`__add__()` (abjad.tools.containertools.Container.Container.Container) (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment) method), 347
`__add__()` (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer) (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment) method), 354
`__add__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory) (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment) method), 382
`__add__()` (abjad.tools.contexttools.Context.Context.Context) (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment) method), 390
`__add__()` (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark) (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass) method), 413
`__add__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory) (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass) method), 415
`__add__()` (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory) (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval) method), 1906
`__add__()` (abjad.tools.durationtools.Duration.Duration.Duration) (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment) method), 447
`__add__()` (abjad.tools.durationtools.Offset.Offset.Offset) (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment) method), 451
`__add__()` (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer) (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval) method), 481
`__add__()` (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory) (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment) method), 659
`__add__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock) (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch) method), 864
`__add__()` (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock) (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass) method), 867
`__add__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile) (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment) method), 876
`__add__()` (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock) (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment) method), 859
`__add__()` (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock) (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch) method), 884
`__add__()` (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory) (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass) method), 938
`__add__()` (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction) (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment) method), 949
`__add__()` (abjad.tools.mathtools.Ratio.Ratio.Ratio) (abjad.tools.pitchtools.ObjectSegment.ObjectSegment.ObjectSegment) method), 954
`__add__()` (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure) (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping) method), 987
`__add__()` (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure) (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory) method), 998
`__add__()` (abjad.tools.measuretools.Measure.Measure.Measure) (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment) method), 1008
`__add__()` (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray) (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment) method), 1051
`__add__()` (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow) (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory) method), 1058
`__add__()` (abjad.tools.pitchtools.Accidental.Accidental.Accidental) (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow) method), 1130

__add__() (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer method), 1334

__add__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1375

__add__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1383

__add__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1392

__add__() (abjad.tools.scoretools.Score.Score.Score method), 1400

__add__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1409

__add__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1418

__add__() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple method), 1432

__add__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1660

__add__() (abjad.tools.stafftools.Staff.Staff.Staff method), 1669

__add__() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 1808

__add__() (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1817

__add__() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1837

__add__() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1846

__add__() (abjad.tools.voicetools.Voice.Voice.Voice method), 1871

__and__() (abjad.tools.chordtools.Chord.Chord.Chord method), 270

__and__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 830

__and__() (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 1035

__and__() (abjad.tools.notetools.Note.Note.Note method), 1040

__and__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1141

__and__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1150

__and__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1156

__and__() (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet method), 1090

__and__() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet method), 1098

__and__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSet.InversionEquivalentChromaticIntervalSet.InversionEquivalentChromaticIntervalSet method), 1162

__and__() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1187

__and__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1201

__and__() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1213

__and__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1219

__and__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1242

__and__() (abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet method), 1112

__and__() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1121

__and__() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet method), 1127

__and__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest method), 1310

__and__() (abjad.tools.resttools.Rest.Rest.Rest method), 1312

__and__() (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass method), 1491

__and__() (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression method), 1493

__and__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1497

__and__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1805

__and__() (abjad.tools.tuplettools.TupletParser.Parser.Parser method), 1880

__call__() (abjad.tools.abjadbooktools.AbjadBookProcessor.AbjadBookProcessor.AbjadBookProcessor method), 1885

__call__() (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.AbjadBookScript method), 1887

__call__() (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock method), 1889

__call__() (abjad.tools.abjadbooktools.HTMLOutputFormat.HTMLOutputFormat.HTMLOutputFormat method), 1890

__call__() (abjad.tools.abjadbooktools.LaTeXOutputFormat.LaTeXOutputFormat.LaTeXOutputFormat method), 1892

__call__() (abjad.tools.abjadbooktools.OutputFormat.OutputFormat.OutputFormat method), 1884

__call__() (abjad.tools.abjadbooktools.ResSetOutputFormat.ResSetOutputFormat.ResSetOutputFormat method), 1893

__call__() (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner method), 228

__call__() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 236

__call__() (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 245

__call__() (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 254

__call__() (abjad.tools.beamtools.MultiplexBeamSpanner.MultiplexBeamSpanner.MultiplexBeamSpanner method), 260

__call__() (abjad.tools.beamtools.NoteLink.NoteLink.NoteLink method), 379

__call__() (abjad.tools.beamtools.ContextMark.ContextMark.ContextMark method), 393

- `__call__()` (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark.abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.
method), 397
- `__call__()` (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark.abjad.tools.documentationtools.Documenter.Documenter.Docum
method), 401
- `__call__()` (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark.abjad.tools.functiontools.FunctionCrawler.FunctionCrawl
method), 404
- `__call__()` (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark.abjad.tools.functiontools.FunctionDocumenter.FunctionD
method), 408
- `__call__()` (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark.abjad.tools.gracetools.GraceContainer.GraceContainer.GraceCo
method), 413
- `__call__()` (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark.abjad.tools.instrumenttools.Accordion.Accordion.Accordion
method), 421
- `__call__()` (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript.abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute
method), 1916
- `__call__()` (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript.abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.Alto
method), 1918
- `__call__()` (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript.abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoT
method), 1921
- `__call__()` (abjad.tools.developerscripttools.CleanScript.CleanScript.CleanScript.abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatCl
method), 1923
- `__call__()` (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScript.abjad.tools.instrumenttools.Bari
method), 1926
- `__call__()` (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScript.abjad.tools.instrumenttools.Baron
method), 1928
- `__call__()` (abjad.tools.developerscripttools.DeveloperScript.DeveloperScript.DeveloperScript.abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClari
method), 1911
- `__call__()` (abjad.tools.developerscripttools.DirectoryScript.DirectoryScript.DirectoryScript.abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute
method), 1913
- `__call__()` (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript.abjad.tools.instru
method), 1930
- `__call__()` (abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.abjad.too
method), 1932
- `__call__()` (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript.RenameModulesScript.abjad.tools.instrumenttools.BassVoice.BassVoice.B
method), 1935
- `__call__()` (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript.abjad.tools.instrumenttools.Bassoon.Bassoon
method), 1938
- `__call__()` (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript.ReplacePromptsScript.abjad.tools.instrumenttools.Cello
method), 1940
- `__call__()` (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScript.abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA
method), 1942
- `__call__()` (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript.abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass
method), 1945
- `__call__()` (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript.abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarin
method), 1947
- `__call__()` (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript.SvnMessageScript.abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.Co
method), 1949
- `__call__()` (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript.abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSa
method), 1952
- `__call__()` (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler.abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Cont
method), 1954
- `__call__()` (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator.abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.Cont
method), 1956
- `__call__()` (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler.abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatCl
method), 1957

__call__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 622

__call__() (abjad.tools.instrumenttools.Flute.Flute.Flute method), 628

__call__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 633

__call__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 639

__call__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 645

__call__() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 651

__call__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 657

__call__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 666

__call__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 672

__call__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 678

__call__() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 684

__call__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 690

__call__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 696

__call__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 702

__call__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 708

__call__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 714

__call__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 720

__call__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 726

__call__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 732

__call__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 738

__call__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 744

__call__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 750

__call__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 756

__call__() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 762

__call__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 768

__call__() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy method), 2035

__call__() (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser method), 2047

__call__() (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser method), 2073

__call__() (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser method), 2077

__call__() (abjad.tools.marktools.Annotation.Annotation.Annotation method), 892

__call__() (abjad.tools.marktools.Articulation.Articulation.Articulation method), 895

__call__() (abjad.tools.marktools.BarLine.BarLine.BarLine method), 899

__call__() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 902

__call__() (abjad.tools.marktoolsDirectedMarkDirectedMarkDirectedMark method), 889

__call__() (abjad.tools.marktoolsLilyPondCommandMarkLilyPondCommandMarkLilyPondCommandMark method), 905

__call__() (abjad.tools.marktoolsLilyPondCommentLilyPondCommentLilyPondComment method), 908

__call__() (abjad.tools.marktoolsMarkMarkMark method), 910

__call__() (abjad.tools.marktoolsStemTremoloStemTremoloStemTremolo method), 913

__call__() (abjad.tools.markuptoolsMarkupMarkupMarkup method), 934

__call__() (abjad.tools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer method), 1334

__call__() (abjad.tools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf method), 1342

__call__() (abjad.tools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode method), 1321

__call__() (abjad.tools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser method), 1345

__call__() (abjad.tools.ScoreTemplatetools.GroupedRhythmicStavesScoreTemplatetools.GroupedRhythmicStavesScoreTemplatetools method), 1361

__call__() (abjad.tools.ScoreTemplatetools.GroupedStavesScoreTemplate.GroupedStavesScoreTemplate method), 1363

__call__() (abjad.tools.ScoreTemplatetools.ScoreTemplate.ScoreTemplate.ScoreTemplate method), 1359

__call__() (abjad.tools.ScoreTemplatetools.StringQuartetScoreTemplate.StringQuartetScoreTemplate method), 1365

__call__() (abjad.tools.ScoreTemplatetools.TwoStaffPianoScoreTemplate.TwoStaffPianoScoreTemplate method), 1367

__call__() (abjad.tools.Spannertools.BracketSpanner.BracketSpanner.BracketSpanner method), 1517

__call__() (abjad.tools.Spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner method), 1524

__call__() (abjad.tools.Spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1533

__call__() (abjad.tools.Spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1541

__call__() (abjad.tools.SpannertoolsDirectedSpannerDirectedSpannerDirectedSpanner method), 1505

__call__() (abjad.tools.SpannertoolsDynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1548

__call__() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.__call__() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner method), 1554

__call__() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.__call__() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1563

__call__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.__call__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner method), 1569

__call__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.__call__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1575

__call__() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.__call__() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner method), 1583

__call__() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.__call__() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner method), 1590

__call__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.__call__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1596

__call__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.__call__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner method), 1603

__call__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.__call__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner method), 1609

__call__() (abjad.tools.spannertools.Spanner.Spanner.Spanner_cmp__() (abjad.tools.spannertools.Spanner.Spanner.Spanner_cmp__() (abjad.tools.spannertools.Spanner.Spanner method), 1510

__call__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.__call__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner method), 1616

__call__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.__call__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner method), 1623

__call__() (abjad.tools.spannertools.TextSpanner.TextSpanner.__call__() (abjad.tools.spannertools.TextSpanner.TextSpanner method), 1629

__call__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.__call__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner method), 1636

__call__() (abjad.tools.tietools.TieSpanner.TieSpanner.__call__() (abjad.tools.tietools.TieSpanner.TieSpanner method), 1689

__call__() (abjad.tools.timetokentools.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker.__call__() (abjad.tools.timetokentools.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker method), 1759

__call__() (abjad.tools.timetokentools.EqualDivisionTimeTokenMaker.EqualDivisionTimeTokenMaker.__call__() (abjad.tools.timetokentools.EqualDivisionTimeTokenMaker.EqualDivisionTimeTokenMaker method), 1769

__call__() (abjad.tools.timetokentools.IncisedTimeTokenMaker.IncisedTimeTokenMaker.__call__() (abjad.tools.timetokentools.IncisedTimeTokenMaker.IncisedTimeTokenMaker method), 1761

__call__() (abjad.tools.timetokentools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.__call__() (abjad.tools.timetokentools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker method), 1772

__call__() (abjad.tools.timetokentools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker.__call__() (abjad.tools.timetokentools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker method), 1775

__call__() (abjad.tools.timetokentools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker.__call__() (abjad.tools.timetokentools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker method), 1779

__call__() (abjad.tools.timetokentools.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker.__call__() (abjad.tools.timetokentools.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker method), 1782

__call__() (abjad.tools.timetokentools.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker.__call__() (abjad.tools.timetokentools.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker method), 1763

__call__() (abjad.tools.timetokentools.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.__call__() (abjad.tools.timetokentools.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker method), 1784

__call__() (abjad.tools.timetokentools.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.__call__() (abjad.tools.timetokentools.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker method), 1788

__call__() (abjad.tools.timetokentools.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker.__call__() (abjad.tools.timetokentools.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker method), 1790

__call__() (abjad.tools.timetokentools.TimeTokenMaker.TimeTokenMaker.__call__() (abjad.tools.timetokentools.TimeTokenMaker.TimeTokenMaker method), 1764

__cmp__() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet method), 1127

__cmp__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionaryTimeIntervalTreeDictionaryTimeIntervalTreeDictionary method), 1743

__cmp__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1805

__complex__() (abjad.tools.durationtools.Duration.Duration.Duration method), 447

__complex__() (abjad.tools.durationtools.Offset.Offset.Offset method), 451

__complex__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction method), 949

__contains__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1881

__contains__() (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner method), 229

__contains__() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 236

__contains__() (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 245

__contains__() (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 254

__contains__() (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 261

__contains__() (abjad.tools.chordtools.Chord.Chord.Chord method), 270

__contains__() (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1897

__contains__() (abjad.tools.containertools.Cluster.Cluster.Cluster method), 340

__contains__() (abjad.tools.containertools.Container.Container.Container method), 347

__contains__() (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer method), 354

__contains__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 382

__contains__() (abjad.tools.contexttools.Context.Context.Context method), 390

__contains__() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 415

__contains__() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method), 1904

__contains__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1906

__contains__() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1967

__contains__() (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 481

__contains__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 660

__contains__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 864

__contains__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 867

__contains__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 876

__contains__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 859

__contains__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 884

__contains__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 938

__contains__() (abjad.tools.mathtools.Ratio.Ratio.Ratio method), 954

__contains__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction method), 987

__contains__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 998

__contains__() (abjad.tools.measuretools.Measure.Measure.Measure method), 1008

__contains__() (abjad.tools.pitchtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 1051

__contains__() (abjad.tools.pitchtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 1136

__contains__() (abjad.tools.pitchtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 1138

__contains__() (abjad.tools.pitchtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 1141

__contains__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1150

__contains__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1153

__contains__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1156

__contains__() (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment.IntervalClassObjectSegment method), 1088

__contains__() (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet method), 1090

__contains__() (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment.IntervalObjectSegment method), 1096

__contains__() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet method), 1098

__contains__() (abjad.tools.pitchtools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 1160

__contains__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment method), 1162

__contains__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1166

__contains__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment method), 1170

__contains__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet method), 1173

__contains__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment method), 1179

__contains__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 1182

__contains__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment method), 1184

- `__contains__()` (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1187
- `__contains__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1198
- `__contains__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1201
- `__contains__()` (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1210
- `__contains__()` (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1213
- `__contains__()` (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment method), 1216
- `__contains__()` (abjad.tools.pitchtools.NamedChromaticPitchSpan.NamedChromaticPitchSpan method), 1219
- `__contains__()` (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector method), 1223
- `__contains__()` (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1238
- `__contains__()` (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1242
- `__contains__()` (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector method), 1245
- `__contains__()` (abjad.tools.pitchtools.ObjectSegment.ObjectSegment method), 1109
- `__contains__()` (abjad.tools.pitchtools.ObjectSet.ObjectSet method), 1112
- `__contains__()` (abjad.tools.pitchtools.ObjectVector.ObjectVector method), 1115
- `__contains__()` (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1252
- `__contains__()` (abjad.tools.pitchtools.OctaveTranspositionMapping.Inventory.OctaveTranspositionMapping.Inventory method), 1258
- `__contains__()` (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment method), 1118
- `__contains__()` (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet method), 1121
- `__contains__()` (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment method), 1125
- `__contains__()` (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet method), 1127
- `__contains__()` (abjad.tools.pitchtools.PitchRange.PitchRange method), 1261
- `__contains__()` (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory method), 1263
- `__contains__()` (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow method), 1267
- `__contains__()` (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer method), 1335
- `__contains__()` (abjad.tools.scoretools.GrandStaff.GrandStaff method), 1375
- `__contains__()` (abjad.tools.scoretools.PerformerInventory.PerformerInventory method), 1383
- `__contains__()` (abjad.tools.scoretools.PianoStaff.PianoStaff method), 1392
- `__contains__()` (abjad.tools.spannertools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1401
- `__contains__()` (abjad.tools.spannertools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1410
- `__contains__()` (abjad.tools.spannertools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1418
- `__contains__()` (abjad.tools.spannertools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1431
- `__contains__()` (abjad.tools.spannertools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1432
- `__contains__()` (abjad.tools.spannertools.NamedChromaticPitchSegment.NamedChromaticPitchSegment method), 1443
- `__contains__()` (abjad.tools.spannertools.NamedChromaticPitchSpan.NamedChromaticPitchSpan method), 1517
- `__contains__()` (abjad.tools.spannertools.NamedChromaticPitchVector.NamedChromaticPitchVector method), 1524
- `__contains__()` (abjad.tools.spannertools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1533
- `__contains__()` (abjad.tools.spannertools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1541
- `__contains__()` (abjad.tools.spannertools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector method), 1505
- `__contains__()` (abjad.tools.spannertools.ObjectSegment.ObjectSegment method), 1548
- `__contains__()` (abjad.tools.spannertools.ObjectSet.ObjectSet method), 1555
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1563
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1569
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1575
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1583
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1590
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1596
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1603
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1609
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1611
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1616
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1623
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1630
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1636
- `__contains__()` (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1661

<code>__contains__()</code> (abjad.tools.stafftools.Staff.Staff.Staff method), 1670	<code>__delattr__()</code> (abjad.tools.exceptiontools.CyclicNodeError.CyclicNodeError method), 1979
<code>__contains__()</code> (abjad.tools.tietools.TieChain.TieChain.TieChain method), 1683	<code>__delattr__()</code> (abjad.tools.exceptiontools.DurationError.DurationError method), 1980
<code>__contains__()</code> (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner method), 1689	<code>__delattr__()</code> (abjad.tools.exceptiontools.ExtraMarkError.ExtraMarkError method), 1981
<code>__contains__()</code> (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1709	<code>__delattr__()</code> (abjad.tools.exceptiontools.ExtraNamedComponentError.ExtraNamedComponentError method), 1982
<code>__contains__()</code> (abjad.tools.timeintervaltools.TimeIntervalTree.FlatIntervalTree.FlatIntervalTree method), 1722	<code>__delattr__()</code> (abjad.tools.exceptiontools.ExtraNoteHeadError.ExtraNoteHeadError method), 1983
<code>__contains__()</code> (abjad.tools.timeintervaltools.TimeIntervalTree.DynamicIntervalTree.DynamicIntervalTree method), 1744	<code>__delattr__()</code> (abjad.tools.exceptiontools.ExtraPitchError.ExtraPitchError method), 1984
<code>__contains__()</code> (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1805	<code>__delattr__()</code> (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError method), 1985
<code>__contains__()</code> (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 1808	<code>__delattr__()</code> (abjad.tools.exceptiontools.GraceContainerError.GraceContainerError method), 1986
<code>__contains__()</code> (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1817	<code>__delattr__()</code> (abjad.tools.exceptiontools.ImpreciseTempoError.ImpreciseTempoError method), 1987
<code>__contains__()</code> (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1837	<code>__delattr__()</code> (abjad.tools.exceptiontools.InputSpecificationError.InputSpecificationError method), 1988
<code>__contains__()</code> (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1847	<code>__delattr__()</code> (abjad.tools.exceptiontools.InstrumentError.InstrumentError method), 1989
<code>__contains__()</code> (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment method), 1860	<code>__delattr__()</code> (abjad.tools.exceptiontools.IntervalError.IntervalError method), 1990
<code>__contains__()</code> (abjad.tools.voicetools.Voice.Voice.Voice method), 1872	<code>__delattr__()</code> (abjad.tools.exceptiontools.LilyPondParserError.LilyPondParserError method), 1991
<code>__del__()</code> (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1971	<code>__delattr__()</code> (abjad.tools.exceptiontools.LineBreakError.LineBreakError method), 1992
<code>__delattr__()</code> (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark method), 379	<code>__delattr__()</code> (abjad.tools.exceptiontools.MarkError.MarkError method), 1993
<code>__delattr__()</code> (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark method), 393	<code>__delattr__()</code> (abjad.tools.exceptiontools.MeasureContiguityError.MeasureContiguityError method), 1994
<code>__delattr__()</code> (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark method), 397	<code>__delattr__()</code> (abjad.tools.exceptiontools.MeasureError.MeasureError method), 1995
<code>__delattr__()</code> (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark method), 401	<code>__delattr__()</code> (abjad.tools.exceptiontools.MissingComponentError.MissingComponentError method), 1996
<code>__delattr__()</code> (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark method), 405	<code>__delattr__()</code> (abjad.tools.exceptiontools.MissingInstrumentError.MissingInstrumentError method), 1997
<code>__delattr__()</code> (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark method), 409	<code>__delattr__()</code> (abjad.tools.exceptiontools.MissingMarkError.MissingMarkError method), 1998
<code>__delattr__()</code> (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method), 413	<code>__delattr__()</code> (abjad.tools.exceptiontools.MissingMeasureError.MissingMeasureError method), 1999
<code>__delattr__()</code> (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark method), 421	<code>__delattr__()</code> (abjad.tools.exceptiontools.MissingNamedComponentError.MissingNamedComponentError method), 2000
<code>__delattr__()</code> (abjad.tools.exceptiontools.AssignabilityError.AssignabilityError.AssignabilityError method), 1974	<code>__delattr__()</code> (abjad.tools.exceptiontools.MissingNoteHeadError.MissingNoteHeadError method), 2001
<code>__delattr__()</code> (abjad.tools.exceptiontools.ClefError.ClefError.ClefError method), 1975	<code>__delattr__()</code> (abjad.tools.exceptiontools.MissingPitchError.MissingPitchError method), 2002
<code>__delattr__()</code> (abjad.tools.exceptiontools.ContainmentError.ContainmentError.ContainmentError method), 1976	<code>__delattr__()</code> (abjad.tools.exceptiontools.MissingSpannerError.MissingSpannerError method), 2003
<code>__delattr__()</code> (abjad.tools.exceptiontools.ContextContainmentError.ContextContainmentError.ContextContainmentError method), 1977	<code>__delattr__()</code> (abjad.tools.exceptiontools.MissingTempoError.MissingTempoError method), 2004
<code>__delattr__()</code> (abjad.tools.exceptiontools.ContiguityError.ContiguityError.ContiguityError method), 1978	<code>__delattr__()</code> (abjad.tools.exceptiontools.MusicContentsError.MusicContentsError method), 2005

__delattr__() (abjad.tools.exceptiontools.NegativeDurationError.NegativeDurationError (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone.
method), 2006

__delattr__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.
method), 2007

__delattr__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.
method), 2008

__delattr__() (abjad.tools.exceptiontools.NoteHeadError.NoteHeadError) (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.
method), 2009

__delattr__() (abjad.tools.exceptiontools.OverfullContainerError.OverfullContainerError) (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.
method), 2010

__delattr__() (abjad.tools.exceptiontools.ParallelError.ParallelError) (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet.
method), 2011

__delattr__() (abjad.tools.exceptiontools.PartitionError.PartitionError) (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute.
method), 2012

__delattr__() (abjad.tools.exceptiontools.PitchError.PitchError) (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.
method), 2013

__delattr__() (abjad.tools.exceptiontools.SchemeParserFinishedException.SchemeParserFinishedException) (abjad.tools.instrumenttools.BassTrombone.BassTrombone.
method), 2014

__delattr__() (abjad.tools.exceptiontools.SpacingError.SpacingError) (abjad.tools.instrumenttools.BassVoice.BassVoice.
method), 2015

__delattr__() (abjad.tools.exceptiontools.SpannerError.SpannerError) (abjad.tools.instrumenttools.Bassoon.Bassoon.
method), 2016

__delattr__() (abjad.tools.exceptiontools.SpannerPopulationError.SpannerPopulationError) (abjad.tools.instrumenttools.Cello.Cello.
method), 2017

__delattr__() (abjad.tools.exceptiontools.StaffContainmentError.StaffContainmentError) (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.
method), 2018

__delattr__() (abjad.tools.exceptiontools.TempoError.TempoError) (abjad.tools.instrumenttools.Contrabass.Contrabass.
method), 2019

__delattr__() (abjad.tools.exceptiontools.TieChainError.TieChainError) (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.
method), 2020

__delattr__() (abjad.tools.exceptiontools.TimeSignatureAssignmentError.TimeSignatureAssignmentError) (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.
method), 2021

__delattr__() (abjad.tools.exceptiontools.TimeSignatureError.TimeSignatureError) (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.
method), 2022

__delattr__() (abjad.tools.exceptiontools.TonalHarmonyError.TonalHarmonyError) (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.
method), 2023

__delattr__() (abjad.tools.exceptiontools.TupletError.TupletError) (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.
method), 2024

__delattr__() (abjad.tools.exceptiontools.TupletFuseError.TupletFuseError) (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.
method), 2025

__delattr__() (abjad.tools.exceptiontools.TypographicWhitespaceError.TypographicWhitespaceError) (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.
method), 2026

__delattr__() (abjad.tools.exceptiontools.UnboundedTimeIntervalError.UnboundedTimeIntervalError) (abjad.tools.instrumenttools.Flute.Flute.
method), 2027

__delattr__() (abjad.tools.exceptiontools.UndefinedSpacingError.UndefinedSpacingError) (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.
method), 2028

__delattr__() (abjad.tools.exceptiontools.UnderfullContainerError.UnderfullContainerError) (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.
method), 2029

__delattr__() (abjad.tools.exceptiontools.VoiceContainmentError.VoiceContainmentError) (abjad.tools.instrumenttools.Guitar.Guitar.
method), 2030

__delattr__() (abjad.tools.instrumenttools.Accordion.Accordion) (abjad.tools.instrumenttools.Harp.Harp.
method), 490

__delattr__() (abjad.tools.instrumenttools.AltoFlute.AltoFlute) (abjad.tools.instrumenttools.Harpsichord.Harpsichord.
method), 496

- `__delattr__()` (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 666
- `__delattr__()` (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 672
- `__delattr__()` (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 678
- `__delattr__()` (abjad.tools.instrumenttools.Piano.Piano.Piano method), 684
- `__delattr__()` (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 690
- `__delattr__()` (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 696
- `__delattr__()` (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 702
- `__delattr__()` (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 708
- `__delattr__()` (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 714
- `__delattr__()` (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 720
- `__delattr__()` (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 726
- `__delattr__()` (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 732
- `__delattr__()` (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 738
- `__delattr__()` (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 744
- `__delattr__()` (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 750
- `__delattr__()` (abjad.tools.instrumenttools.Viola.Viola.Viola method), 756
- `__delattr__()` (abjad.tools.instrumenttools.Violin.Violin.Violin method), 762
- `__delattr__()` (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 768
- `__delattr__()` (abjad.tools.marktools.Annotation.Annotation.Annotation method), 892
- `__delattr__()` (abjad.tools.marktools.Articulation.Articulation.Articulation method), 895
- `__delattr__()` (abjad.tools.marktools.BarLine.BarLine.BarLine method), 899
- `__delattr__()` (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 902
- `__delattr__()` (abjad.tools.marktools.DirectedMark.DirectedMark.DirectedMark method), 889
- `__delattr__()` (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 905
- `__delattr__()` (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method), 908
- `__delattr__()` (abjad.tools.marktools.Mark.Mark.Mark method), 910
- `__delattr__()` (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method), 913
- `__delitem__()` (abjad.tools.markuptools.Markup.Markup.Markup method), 934
- `__delitem__()` (abjad.tools.markuptools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 270
- `__delitem__()` (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1897
- `__delitem__()` (abjad.tools.containertools.Cluster.Cluster.Cluster method), 340
- `__delitem__()` (abjad.tools.containertools.Container.Container.Container method), 347
- `__delitem__()` (abjad.tools.containertools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 354
- `__delitem__()` (abjad.tools.clefmarkinventory.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 382
- `__delitem__()` (abjad.tools.contexttools.Context.Context.Context method), 390
- `__delitem__()` (abjad.tools.temptomarkinventory.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 416
- `__delitem__()` (abjad.tools.immutabledictionary.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method), 1904
- `__delitem__()` (abjad.tools.datastructretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1906
- `__delitem__()` (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1967
- `__delitem__()` (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 481
- `__delitem__()` (abjad.tools.instrumentinventory.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 660
- `__delitem__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 864
- `__delitem__()` (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 867
- `__delitem__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 876
- `__delitem__()` (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 859
- `__delitem__()` (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 884
- `__delitem__()` (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 938
- `__delitem__()` (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 987
- `__delitem__()` (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 998
- `__delitem__()` (abjad.tools.measuretools.Measure.Measure.Measure method), 1008
- `__delitem__()` (abjad.tools.lilypondfiletools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 1136
- `__delitem__()` (abjad.tools.lilypondfiletools.LilyPondComment.LilyPondComment.LilyPondComment method), 1166
- `__delitem__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector method), 1173
- `__delitem__()` (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 1182

__delitem__() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector method), 1223

__delitem__() (abjad.tools.pitchtools.NumberedChromaticPitchVector.NumberedChromaticPitchVector.NumberedChromaticPitchVector method), 1245

__delitem__() (abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector method), 1115

__delitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1252

__delitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1258

__delitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1263

__delitem__() (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer method), 1335

__delitem__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1375

__delitem__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1383

__delitem__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1392

__delitem__() (abjad.tools.scoretools.Score.Score.Score __div__() (abjad.tools.durationtools.Offset.Offset.Offset method), 1401

__delitem__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1410

__delitem__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1418

__delitem__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1661

__delitem__() (abjad.tools.stafftools.Staff.Staff.Staff __divmod__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction method), 1670

__delitem__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1709

__delitem__() (abjad.tools.timeintervaltools.TimeIntervalTree.Dictionary(Dictionary.Dictionary method), 1744

__delitem__() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1837

__delitem__() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet __eq__() (abjad.tools.abctools.Parser.Parser.Parser method), 1847

__delitem__() (abjad.tools.voicetools.Voice.Voice.Voice __eq__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1872

__delslice__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 382

__delslice__() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 416

__delslice__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1906

__delslice__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 660

__delslice__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 864

__delslice__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 867

__delslice__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 876

__delitem__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction method), 950

__delitem__() (abjad.tools.durationtools.Duration.Duration.Duration method), 447

__delitem__() (abjad.tools.durationtools.Offset.Offset.Offset method), 451

__delitem__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction method), 950

__delitem__() (abjad.tools.abctools.Parser.Parser.Parser method), 1880

__delitem__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1881

__delitem__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1882

__delitem__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1885

__delitem__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1887

__delitem__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1889

__delitem__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1890

__delitem__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1892

__delitem__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1884

- [__eq__\(\) \(abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutputFormat.OrdinalConstant.OrdinalConstant.C method\), 1894](#)
- [__eq__\(\) \(abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner method\), 229](#)
- [__eq__\(\) \(abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method\), 236](#)
- [__eq__\(\) \(abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method\), 245](#)
- [__eq__\(\) \(abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method\), 254](#)
- [__eq__\(\) \(abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method\), 261](#)
- [__eq__\(\) \(abjad.tools.chordtools.Chord.Chord.Chord method\), 270](#)
- [__eq__\(\) \(abjad.tools.componenttools.Component.Component.Component method\), 277](#)
- [__eq__\(\) \(abjad.tools.componenttools.ContainmentSignature.ContainmentSignature.ContainmentSignature method\), 278](#)
- [__eq__\(\) \(abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method\), 1897](#)
- [__eq__\(\) \(abjad.tools.containertools.Cluster.Cluster.Cluster method\), 340](#)
- [__eq__\(\) \(abjad.tools.containertools.Container.Container.Container method\), 347](#)
- [__eq__\(\) \(abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer method\), 355](#)
- [__eq__\(\) \(abjad.tools.contexttools.ClefMark.ClefMark.ClefMark method\), 379](#)
- [__eq__\(\) \(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method\), 382](#)
- [__eq__\(\) \(abjad.tools.contexttools.Context.Context.Context method\), 390](#)
- [__eq__\(\) \(abjad.tools.contexttools.ContextMark.ContextMark.ContextMark method\), 393](#)
- [__eq__\(\) \(abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark method\), 397](#)
- [__eq__\(\) \(abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark method\), 401](#)
- [__eq__\(\) \(abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark method\), 405](#)
- [__eq__\(\) \(abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark method\), 409](#)
- [__eq__\(\) \(abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method\), 413](#)
- [__eq__\(\) \(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method\), 416](#)
- [__eq__\(\) \(abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark method\), 421](#)
- [__eq__\(\) \(abjad.tools.datastructuretools.Digraph.Digraph.Digraph method\), 1901](#)
- [__eq__\(\) \(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method\), 1904](#)
- [__eq__\(\) \(abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method\), 1906](#)
- [__eq__\(\) \(abjad.tools.developmenttools.AbjDevScript.AbjDevScript.AbjDevScript method\), 1916](#)
- [__eq__\(\) \(abjad.tools.developmenttools.AbjGrepScript.AbjGrepScript.AbjGrepScript method\), 1919](#)
- [__eq__\(\) \(abjad.tools.developmenttools.AbjMakeScript.AbjMakeScript.AbjMakeScript method\), 1921](#)
- [__eq__\(\) \(abjad.tools.developmenttools.AbjMakeScript.AbjMakeScript.AbjMakeScript method\), 1923](#)
- [__eq__\(\) \(abjad.tools.developmenttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScript method\), 1926](#)
- [__eq__\(\) \(abjad.tools.developmenttools.CountToolsScript.CountToolsScript.CountToolsScript method\), 1928](#)
- [__eq__\(\) \(abjad.tools.developmenttools.DeveloperScript.DeveloperScript.DeveloperScript method\), 1911](#)
- [__eq__\(\) \(abjad.tools.developmenttools.DirectoryScript.DirectoryScript.DirectoryScript method\), 1913](#)
- [__eq__\(\) \(abjad.tools.developmenttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript method\), 1930](#)
- [__eq__\(\) \(abjad.tools.developmenttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript method\), 1932](#)
- [__eq__\(\) \(abjad.tools.developmenttools.RenameModulesScript.RenameModulesScript.RenameModulesScript method\), 1935](#)
- [__eq__\(\) \(abjad.tools.developmenttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript method\), 1938](#)
- [__eq__\(\) \(abjad.tools.developmenttools.ReplacePromptsScript.ReplacePromptsScript.ReplacePromptsScript method\), 1940](#)
- [__eq__\(\) \(abjad.tools.developmenttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScript method\), 1943](#)
- [__eq__\(\) \(abjad.tools.developmenttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript method\), 1945](#)
- [__eq__\(\) \(abjad.tools.developmenttools.SvnCommitScript.SvnCommitScript.SvnCommitScript method\), 1947](#)
- [__eq__\(\) \(abjad.tools.developmenttools.SvnMessageScript.SvnMessageScript.SvnMessageScript method\), 1950](#)
- [__eq__\(\) \(abjad.tools.developmenttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript method\), 1952](#)
- [__eq__\(\) \(abjad.tools.developmenttools.AbjadKeySignatureMark.AbjadKeySignatureMark.AbjadKeySignatureMark method\), 1954](#)
- [__eq__\(\) \(abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator method\), 1956](#)
- [__eq__\(\) \(abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler method\), 1957](#)
- [__eq__\(\) \(abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter method\), 1959](#)
- [__eq__\(\) \(abjad.tools.documentationtools.Documenter.Documenter.Documenter method\), 1961](#)
- [__eq__\(\) \(abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler method\), 1962](#)
- [__eq__\(\) \(abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter method\), 1964](#)
- [__eq__\(\) \(abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method\), 1967](#)

[__eq__\(\) \(abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler method\), 1968](#)
[__eq__\(\) \(abjad.tools.documentationtools.Pipe.Pipe.Pipe method\), 1971](#)
[__eq__\(\) \(abjad.tools.durationtools.Duration.Duration.Duration method\), 447](#)
[__eq__\(\) \(abjad.tools.durationtools.Offset.Offset.Offset method\), 451](#)
[__eq__\(\) \(abjad.tools.durationtools.TimespanConstant.TimespanConstant.TimespanConstant method\), 454](#)
[__eq__\(\) \(abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method\), 481](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Accordion.Accordion.Accordion method\), 490](#)
[__eq__\(\) \(abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method\), 496](#)
[__eq__\(\) \(abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone method\), 502](#)
[__eq__\(\) \(abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method\), 508](#)
[__eq__\(\) \(abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method\), 514](#)
[__eq__\(\) \(abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone method\), 520](#)
[__eq__\(\) \(abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice method\), 526](#)
[__eq__\(\) \(abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method\), 532](#)
[__eq__\(\) \(abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method\), 538](#)
[__eq__\(\) \(abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method\), 544](#)
[__eq__\(\) \(abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method\), 550](#)
[__eq__\(\) \(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method\), 556](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method\), 562](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Cello.Cello.Cello method\), 568](#)
[__eq__\(\) \(abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method\), 574](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method\), 580](#)
[__eq__\(\) \(abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method\), 586](#)
[__eq__\(\) \(abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method\), 592](#)
[__eq__\(\) \(abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method\), 598](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method\), 604](#)
[__eq__\(\) \(abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method\), 610](#)
[__eq__\(\) \(abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method\), 616](#)
[__eq__\(\) \(abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method\), 622](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Flute.Flute.Flute method\), 628](#)
[__eq__\(\) \(abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method\), 633](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method\), 639](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Guitar.Guitar.Guitar method\), 645](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Harp.Harp.Harp method\), 651](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method\), 657](#)
[__eq__\(\) \(abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method\), 660](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Marimba.Marimba.Marimba method\), 666](#)
[__eq__\(\) \(abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method\), 672](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Oboe.Oboe.Oboe method\), 678](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Piano.Piano.Piano method\), 684](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method\), 690](#)
[__eq__\(\) \(abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method\), 696](#)
[__eq__\(\) \(abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method\), 702](#)
[__eq__\(\) \(abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method\), 708](#)
[__eq__\(\) \(abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method\), 714](#)
[__eq__\(\) \(abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method\), 720](#)
[__eq__\(\) \(abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method\), 726](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method\), 732](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Tuba.Tuba.Tuba method\), 738](#)
[__eq__\(\) \(abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method\), 744](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method\), 750](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Viola.Viola.Viola method\), 756](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Violin.Violin.Violin method\), 762](#)
[__eq__\(\) \(abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method\), 768](#)

<code>__eq__()</code> (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 824	<code>__eq__()</code> (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method), 908
<code>__eq__()</code> (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 830	<code>__eq__()</code> (abjad.tools.marktools.Mark.Mark.Mark method), 910
<code>__eq__()</code> (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 862	<code>__eq__()</code> (abjad.tools.marktools.StemToken.StemToken.StemToken method), 913
<code>__eq__()</code> (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 864	<code>__eq__()</code> (abjad.tools.markuptools.Markup.Markup.Markup method), 934
<code>__eq__()</code> (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 867	<code>__eq__()</code> (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand method), 936
<code>__eq__()</code> (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken method), 871	<code>__eq__()</code> (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 938
<code>__eq__()</code> (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 876	<code>__eq__()</code> (abjad.tools.mathtools.BoundedObject.BoundedObject.BoundedObject method), 946
<code>__eq__()</code> (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 879	<code>__eq__()</code> (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction method), 950
<code>__eq__()</code> (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken method), 880	<code>__eq__()</code> (abjad.tools.mathtools.Ratio.Ratio.Ratio method), 954
<code>__eq__()</code> (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 860	<code>__eq__()</code> (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 987
<code>__eq__()</code> (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 885	<code>__eq__()</code> (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 998
<code>__eq__()</code> (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy method), 2035	<code>__eq__()</code> (abjad.tools.measuretools.Measure.Measure.Measure method), 1008
<code>__eq__()</code> (abjad.tools.lilypondparsertools.LilyPondDuration.LilyPondDuration.LilyPondDuration method), 2037	<code>__eq__()</code> (abjad.tools.mathtools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 1035
<code>__eq__()</code> (abjad.tools.lilypondparsertools.LilyPondEvent.LilyPondEvent.LilyPondEvent method), 2038	<code>__eq__()</code> (abjad.tools.mathtools.Note.Note.Note method), 1040
<code>__eq__()</code> (abjad.tools.lilypondparsertools.LilyPondFraction.LilyPondFraction.LilyPondFraction method), 2039	<code>__eq__()</code> (abjad.tools.mathtools.NoteHead.NoteHead.NoteHead method), 1042
<code>__eq__()</code> (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition.LilyPondLexicalDefinition method), 2043	<code>__eq__()</code> (abjad.tools.mathtools.PitchArray.PitchArray.PitchArray method), 1051
<code>__eq__()</code> (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser method), 2047	<code>__eq__()</code> (abjad.tools.mathtools.PitchArrayCell.PitchArrayCell.PitchArrayCell method), 1053
<code>__eq__()</code> (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2066	<code>__eq__()</code> (abjad.tools.mathtools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1056
<code>__eq__()</code> (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser method), 2073	<code>__eq__()</code> (abjad.tools.mathtools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1058
<code>__eq__()</code> (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser method), 2077	<code>__eq__()</code> (abjad.tools.mathtools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1130
<code>__eq__()</code> (abjad.tools.lilypondparsertools.SyntaxNode.SyntaxNode.SyntaxNode method), 2078	<code>__eq__()</code> (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject.ChromaticIntervalClassObject method), 1064
<code>__eq__()</code> (abjad.tools.marktools.Annotation.Annotation.Annotation method), 892	<code>__eq__()</code> (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.ChromaticIntervalObject method), 1066
<code>__eq__()</code> (abjad.tools.marktools.Articulation.Articulation.Articulation method), 895	<code>__eq__()</code> (abjad.tools.pitchtools.ChromaticObject.ChromaticObject.ChromaticObject method), 1068
<code>__eq__()</code> (abjad.tools.marktools.BarLine.BarLine.BarLine method), 899	<code>__eq__()</code> (abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject.ChromaticPitchObject method), 1069
<code>__eq__()</code> (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 902	<code>__eq__()</code> (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject.CounterpointIntervalClassObject method), 1071
<code>__eq__()</code> (abjad.tools.marktools.DirectedMark.DirectedMark.DirectedMark method), 889	<code>__eq__()</code> (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.CounterpointIntervalObject method), 1072
<code>__eq__()</code> (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 905	<code>__eq__()</code> (abjad.tools.pitchtools.CounterpointObject.CounterpointObject.CounterpointObject method), 1074

Index 2125

- `__eq__()` (abjad.tools.timetoken.tools.TupletMonadTimeToken.Maker.TupletMonadWellFormedTokenMaker.EuphonicMonadTimeToken.NakedMethod), 1801
- `__eq__()` (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass(abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck)), 1805
- `__eq__()` (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator(abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck)), 1808
- `__eq__()` (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator(abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck)), 1810
- `__eq__()` (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator(abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck)), 1811
- `__eq__()` (abjad.tools.tonalitytools.Mode.Mode.Mode `__float__()` (abjad.tools.durationtools.Duration.Duration.Duration)), 1812
- `__eq__()` (abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator(abjad.tools.wellformednesstools.Offset.Offset.Offset)), 1813
- `__eq__()` (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.QualityIndicator(abjad.tools.wellformednesstools.NonreducedFraction.NonreducedFraction)), 1815
- `__eq__()` (abjad.tools.tonalitytools.Scale.Scale.Scale `__float__()` (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject)), 1817
- `__eq__()` (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree(abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject)), 1819
- `__eq__()` (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator(abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject)), 1821
- `__eq__()` (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction(abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject)), 1822
- `__eq__()` (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet(abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject)), 1838
- `__eq__()` (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet `__float__()` (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject)), 1847
- `__eq__()` (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment(abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject)), 1860
- `__eq__()` (abjad.tools.voicetools.Voice.Voice.Voice `__float__()` (abjad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClassObject)), 1872
- `__eq__()` (abjad.tools.wellformednesstools.BeamedQuarterNoteCheck(BeamedQuarterNoteCheck)), 2082
- `__eq__()` (abjad.tools.wellformednesstools.Check.Check.Check `__float__()` (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval)), 2081
- `__eq__()` (abjad.tools.wellformednesstools.DiscontiguousSpanCheck(DiscontiguousSpanCheck)), 2084
- `__eq__()` (abjad.tools.wellformednesstools.DuplicateIdCheck(DuplicateIdCheck)), 2085
- `__eq__()` (abjad.tools.wellformednesstools.EmptyContainerCheck(EmptyContainerCheck)), 2087
- `__eq__()` (abjad.tools.wellformednesstools.IntermarkedHairpinCheck(IntermarkedHairpinCheck)), 2088
- `__eq__()` (abjad.tools.wellformednesstools.MisduratedMeasureCheck(MisduratedMeasureCheck)), 2090
- `__eq__()` (abjad.tools.wellformednesstools.MisfilledMeasureCheck(MisfilledMeasureCheck)), 2091
- `__eq__()` (abjad.tools.wellformednesstools.MispitchedTieCheck(MispitchedTieCheck)), 2093
- `__eq__()` (abjad.tools.wellformednesstools.MisrepresentedFlagCheck(MisrepresentedFlagCheck)), 2094
- `__eq__()` (abjad.tools.wellformednesstools.MissingParentCheck(MissingParentCheck)), 2096

- __ge__() (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.MahJahKeySignatureMarkiontools.APICrawler.APICrawler.APICrawler method), 405
- __ge__() (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark.abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator method), 409
- __ge__() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark.abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler method), 413
- __ge__() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter method), 416
- __ge__() (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark.abjad.tools.documentationtools.Documenter.Documenter.Documenter method), 421
- __ge__() (abjad.tools.datastructuretools.Digraph.Digraph.Digraph) (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler method), 1901
- __ge__() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary.abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter method), 1904
- __ge__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1906
- __ge__() (abjad.tools.datastructuretools.OrdinalConstant.OrdinalConstant.OrdinalConstant.abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler method), 1909
- __ge__() (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript.abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1916
- __ge__() (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript.abjad.tools.documentationtools.Duration.Duration.Duration method), 1919
- __ge__() (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript.abjad.tools.documentationtools.Offset.Offset.Offset method), 1921
- __ge__() (abjad.tools.developerscripttools.CleanScript.CleanScript.CleanScript.abjad.tools.durationtools.TimespanConstant.TimespanConstant.TimespanConstant method), 1923
- __ge__() (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScript.abjad.tools.durationtools.Container.GraceContainer.GraceContainer method), 1926
- __ge__() (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScript.abjad.tools.durationtools.Accordion.Accordion.Accordion method), 1928
- __ge__() (abjad.tools.developerscripttools.DeveloperScript.DeveloperScript.DeveloperScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1911
- __ge__() (abjad.tools.developerscripttools.DirectoryScript.DirectoryScript.DirectoryScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1913
- __ge__() (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1930
- __ge__() (abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1932
- __ge__() (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript.RenameModulesScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1935
- __ge__() (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1938
- __ge__() (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript.ReplacePromptsScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1940
- __ge__() (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1943
- __ge__() (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1945
- __ge__() (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1947
- __ge__() (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript.SvnMessageScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1950
- __ge__() (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript.abjad.tools.altoflute.AltoFlute.AltoFlute method), 1952

- `__ge__()` (abjad.tools.instrumenttools.Cello.Cello.Cello method), 568
- `__ge__()` (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 574
- `__ge__()` (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 580
- `__ge__()` (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 586
- `__ge__()` (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 592
- `__ge__()` (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 598
- `__ge__()` (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 604
- `__ge__()` (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 610
- `__ge__()` (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 616
- `__ge__()` (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 622
- `__ge__()` (abjad.tools.instrumenttools.Flute.Flute.Flute method), 628
- `__ge__()` (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 633
- `__ge__()` (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 639
- `__ge__()` (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 645
- `__ge__()` (abjad.tools.instrumenttools.Harp.Harp.Harp method), 651
- `__ge__()` (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 657
- `__ge__()` (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 660
- `__ge__()` (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 666
- `__ge__()` (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 672
- `__ge__()` (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 678
- `__ge__()` (abjad.tools.instrumenttools.Piano.Piano.Piano method), 684
- `__ge__()` (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 690
- `__ge__()` (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 696
- `__ge__()` (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 702
- `__ge__()` (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 708
- `__ge__()` (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 714
- `__ge__()` (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 720
- `__ge__()` (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 726
- `__ge__()` (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 732
- `__ge__()` (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 738
- `__ge__()` (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 744
- `__ge__()` (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 750
- `__ge__()` (abjad.tools.instrumenttools.Viola.Viola.Viola method), 756
- `__ge__()` (abjad.tools.instrumenttools.Violin.Violin.Violin method), 762
- `__ge__()` (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 768
- `__ge__()` (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 824
- `__ge__()` (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 830
- `__ge__()` (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken method), 862
- `__ge__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 864
- `__ge__()` (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 867
- `__ge__()` (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken method), 871
- `__ge__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 876
- `__ge__()` (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken method), 879
- `__ge__()` (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken method), 880
- `__ge__()` (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 860
- `__ge__()` (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 885
- `__ge__()` (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy method), 2036
- `__ge__()` (abjad.tools.lilypondparsertools.LilyPondDuration.LilyPondDuration.LilyPondDuration method), 2037
- `__ge__()` (abjad.tools.lilypondparsertools.LilyPondEvent.LilyPondEvent.LilyPondEvent method), 2038
- `__ge__()` (abjad.tools.lilypondparsertools.LilyPondFraction.LilyPondFraction.LilyPondFraction method), 2039
- `__ge__()` (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition.LilyPondLexicalDefinition method), 2043
- `__ge__()` (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser method), 2047
- `__ge__()` (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2066
- `__ge__()` (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser method), 2073

<code>__ge__()</code> (abjad.tools.lilypondparsertools.SchemeParser.SchemeParserSchemeParser method), 2077	<code>__ge__()</code> (abjad.tools.pitchtools.Accidental.Accidental.Accidental method), 1130
<code>__ge__()</code> (abjad.tools.lilypondparsertools.SyntaxNode.SyntaxNode.SyntaxNode method), 2078	<code>__ge__()</code> (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject method), 1065
<code>__ge__()</code> (abjad.tools.marktools.Annotation.Annotation.Annotation method), 892	<code>__ge__()</code> (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject method), 1066
<code>__ge__()</code> (abjad.tools.marktools.Articulation.Articulation.Articulation method), 895	<code>__ge__()</code> (abjad.tools.pitchtools.ChromaticObject.ChromaticObject.ChromaticObject method), 1068
<code>__ge__()</code> (abjad.tools.marktools.BarLine.BarLine.BarLine method), 899	<code>__ge__()</code> (abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject method), 1069
<code>__ge__()</code> (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 902	<code>__ge__()</code> (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject method), 1071
<code>__ge__()</code> (abjad.tools.marktools.DirectedMark.DirectedMark.DirectedMark method), 889	<code>__ge__()</code> (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject method), 1072
<code>__ge__()</code> (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 905	<code>__ge__()</code> (abjad.tools.pitchtools.CounterpointObject.CounterpointObject.CounterpointObject method), 1074
<code>__ge__()</code> (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method), 908	<code>__ge__()</code> (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject method), 1075
<code>__ge__()</code> (abjad.tools.marktools.Mark.Mark.Mark method), 910	<code>__ge__()</code> (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject method), 1077
<code>__ge__()</code> (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method), 913	<code>__ge__()</code> (abjad.tools.pitchtools.DiatonicObject.DiatonicObject.DiatonicObject method), 1078
<code>__ge__()</code> (abjad.tools.markuptools.Markup.Markup.Markup method), 934	<code>__ge__()</code> (abjad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClassObject method), 1080
<code>__ge__()</code> (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand method), 936	<code>__ge__()</code> (abjad.tools.pitchtools.DiatonicPitchObject.DiatonicPitchObject.DiatonicPitchObject method), 1081
<code>__ge__()</code> (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 938	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval method), 1132
<code>__ge__()</code> (abjad.tools.mathtools.BoundedObject.BoundedObject.BoundedObject method), 946	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 1133
<code>__ge__()</code> (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction method), 950	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1136
<code>__ge__()</code> (abjad.tools.mathtools.Ratio.Ratio.Ratio method), 954	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1138
<code>__ge__()</code> (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 987	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1141
<code>__ge__()</code> (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 998	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 1143
<code>__ge__()</code> (abjad.tools.measuretools.Measure.Measure.Measure method), 1008	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1144
<code>__ge__()</code> (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 1035	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval method), 1146
<code>__ge__()</code> (abjad.tools.notetools.Note.Note.Note method), 1040	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1148
<code>__ge__()</code> (abjad.tools.notetools.NoteHead.NoteHead.NoteHead method), 1042	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1150
<code>__ge__()</code> (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method), 1051	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1153
<code>__ge__()</code> (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell method), 1053	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1156
<code>__ge__()</code> (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1056	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject method), 1083
<code>__ge__()</code> (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1058	<code>__ge__()</code> (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject method), 1084

[__ge__\(\) \(abjad.tools.pitchtools.HarmonicObject.HarmonicObject.IntervalObject.IntervalObject.pitchtools.MelodicIntervalObject.MelodicIntervalObject method\), 1086](#)
[__ge__\(\) \(abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment.IntervalClassObjectSegment.MelodicIntervalObject.MelodicIntervalObject method\), 1088](#)
[__ge__\(\) \(abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet.NamedChromaticPitch.NamedChromaticPitch method\), 1090](#)
[__ge__\(\) \(abjad.tools.pitchtools.IntervalObject.IntervalObject.IntervalObject.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass method\), 1092](#)
[__ge__\(\) \(abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass.IntervalObjectClass.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method\), 1094](#)
[__ge__\(\) \(abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment.IntervalObjectSegment.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method\), 1096](#)
[__ge__\(\) \(abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet.NamedChromaticPitchSegment.NamedChromaticPitchSegment method\), 1098](#)
[__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.NamedChromaticIntervalClass.NamedChromaticIntervalClass method\), 1158](#)
[__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment.NamedChromaticIntervalClassSegment.NamedChromaticIntervalClassSegment method\), 1160](#)
[__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.NamedChromaticIntervalClassSet.NamedChromaticIntervalClassSet method\), 1163](#)
[__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalVector.InversionEquivalentChromaticIntervalVector.NamedChromaticIntervalVector.NamedChromaticIntervalVector method\), 1166](#)
[__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.NamedDiatonicIntervalClass.NamedDiatonicIntervalClass method\), 1168](#)
[__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment.NamedDiatonicIntervalClassSegment.NamedDiatonicIntervalClassSegment method\), 1170](#)
[__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet.NamedDiatonicIntervalClassSet.NamedDiatonicIntervalClassSet method\), 1173](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.NamedChromaticInterval.NamedChromaticInterval method\), 1175](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.NamedChromaticIntervalClass.NamedChromaticIntervalClass method\), 1177](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.NamedChromaticIntervalClassSegment.NamedChromaticIntervalClassSegment method\), 1179](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.NamedChromaticIntervalClassVector.NamedChromaticIntervalClassVector method\), 1182](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.NamedChromaticIntervalSegment.NamedChromaticIntervalSegment method\), 1184](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.NamedChromaticIntervalSet.NamedChromaticIntervalSet method\), 1187](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.NamedCounterpointInterval.NamedCounterpointInterval method\), 1190](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.NamedCounterpointIntervalClass.NamedCounterpointIntervalClass method\), 1191](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.NamedDiatonicInterval.NamedDiatonicInterval method\), 1194](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.NamedDiatonicIntervalClass.NamedDiatonicIntervalClass method\), 1196](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.NamedDiatonicIntervalSegment.NamedDiatonicIntervalSegment method\), 1198](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.NamedDiatonicIntervalSet.NamedDiatonicIntervalSet method\), 1201](#)
[__ge__\(\) \(abjad.tools.pitchtools.MelodicIntervalClassObject.MelodicIntervalClassObject.NamedIntervalClassObject.NamedIntervalClassObject method\), 1100](#)

- `__ge__()` (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1258
- `__ge__()` (abjad.tools.pitchtools.PitchClassObject.PitchClassObject.PitchClassObject method), 1367
- `__ge__()` (abjad.tools.pitchtools.PitchClassObject.PitchClassObject.PitchClassObject method), 1375
- `__ge__()` (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment.PitchClassObjectSegment method), 1379
- `__ge__()` (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1381
- `__ge__()` (abjad.tools.pitchtools.PitchObject.PitchObject.PitchObject method), 1383
- `__ge__()` (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment.PitchObjectSegment method), 1392
- `__ge__()` (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet method), 1401
- `__ge__()` (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange method), 1410
- `__ge__()` (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1419
- `__ge__()` (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 1422
- `__ge__()` (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest method), 1431
- `__ge__()` (abjad.tools.resttools.Rest.Rest.Rest method), `__ge__()` (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple method), 1432
- `__ge__()` (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer method), 1435
- `__ge__()` (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf method), 1443
- `__ge__()` (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode method), 1491
- `__ge__()` (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser method), 1493
- `__ge__()` (abjad.tools.schemetools.Scheme.Scheme.Scheme method), `__ge__()` (abjad.tools.skiptools.Skip.Skip.Skip method), 1497
- `__ge__()` (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList method), 1518
- `__ge__()` (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor method), 1524
- `__ge__()` (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment method), 1533
- `__ge__()` (abjad.tools.schemetools.SchemePair.SchemePair.SchemePair method), 1541
- `__ge__()` (abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector method), 1505
- `__ge__()` (abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant.SchemeVectorConstant method), 1548
- `__ge__()` (abjad.tools.scoretemplatetools.GroupedRhythmicStavesScoreTemplate.GroupedRhythmicStavesScoreTemplate.GroupedRhythmicStavesScoreTemplate method), 1555
- `__ge__()` (abjad.tools.scoretemplatetools.GroupedStavesScoreTemplate.GroupedStavesScoreTemplate.GroupedStavesScoreTemplate method), 1563
- `__ge__()` (abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplate.ScoreTemplate method), 1569
- `__ge__()` (abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.StringQuartetScoreTemplate.StringQuartetScoreTemplate method), 1575

- `__ge__()` (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2085
- `__ge__()` (abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2087
- `__ge__()` (abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2088
- `__ge__()` (abjad.tools.wellformednesstools.MisduratedMeasureCheck.MisduratedMeasureCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2090
- `__ge__()` (abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2091
- `__ge__()` (abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2093
- `__ge__()` (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2094
- `__ge__()` (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2096
- `__ge__()` (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2097
- `__ge__()` (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2099
- `__ge__()` (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2100
- `__ge__()` (abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2102
- `__ge__()` (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary method), 2103
- `__getitem__()` (abjad.tools.abctools.ScoreSelection.ScoreSelection.abjad.tools.abctools.ScoreSelection method), 1881
- `__getitem__()` (abjad.tools.beamttools.BeamSpanner.BeamSpanner.abjad.tools.beamttools.BeamSpanner method), 229
- `__getitem__()` (abjad.tools.beamttools.ComplexBeamSpanner.ComplexBeamSpanner.abjad.tools.beamttools.ComplexBeamSpanner method), 237
- `__getitem__()` (abjad.tools.beamttools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.abjad.tools.beamttools.DuratedComplexBeamSpanner method), 245
- `__getitem__()` (abjad.tools.beamttools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.abjad.tools.beamttools.MeasuredComplexBeamSpanner method), 254
- `__getitem__()` (abjad.tools.beamttools.MultipartBeamSpanner.MultipartBeamSpanner.abjad.tools.beamttools.MultipartBeamSpanner method), 261
- `__getitem__()` (abjad.tools.chordtools.Chord.Chord.Chord method), 270
- `__getitem__()` (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.abjad.tools.configurationtools.AbjadConfig method), 1897
- `__getitem__()` (abjad.tools.containertools.Cluster.Cluster.Cluster method), 341
- `__getitem__()` (abjad.tools.containertools.Container.Container.Container method), 347
- `__getitem__()` (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.abjad.tools.containertools.FixedDurationContainer method), 355
- `__getitem__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.abjad.tools.contexttools.ClefMarkInventory method), 382
- `__getitem__()` (abjad.tools.contexttools.Context.Context.Context method), 391
- `__getitem__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.abjad.tools.contexttools.TempoMarkInventory method), 416
- `__getitem__()` (abjad.tools.exceptiontools.ExtraPitchError.ExtraPitchError.abjad.tools.exceptiontools.ExtraPitchError method), 1884
- `__getitem__()` (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.abjad.tools.exceptiontools.ExtraSpannerError method), 1885
- `__getitem__()` (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.abjad.tools.exceptiontools.ExtraSpannerError method), 1886
- `__getitem__()` (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.abjad.tools.exceptiontools.ExtraSpannerError method), 1887
- `__getitem__()` (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.abjad.tools.exceptiontools.ExtraSpannerError method), 1888
- `__getitem__()` (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.abjad.tools.exceptiontools.ExtraSpannerError method), 1889
- `__getitem__()` (abjad.tools.exceptiontools.IntervalError.IntervalError.abjad.tools.exceptiontools.IntervalError method), 1900
- `__getitem__()` (abjad.tools.exceptiontools.LilyPondParserError.LilyPondParserError.abjad.tools.exceptiontools.LilyPondParserError method), 1991
- `__getitem__()` (abjad.tools.exceptiontools.LineBreakError.LineBreakError.abjad.tools.exceptiontools.LineBreakError method), 1992
- `__getitem__()` (abjad.tools.exceptiontools.MarkError.MarkError.abjad.tools.exceptiontools.MarkError method), 1993
- `__getitem__()` (abjad.tools.exceptiontools.MeasureError.MeasureError.abjad.tools.exceptiontools.MeasureError method), 1994
- `__getitem__()` (abjad.tools.exceptiontools.MeasureError.MeasureError.abjad.tools.exceptiontools.MeasureError method), 1995
- `__getitem__()` (abjad.tools.exceptiontools.MissingComponentError.MissingComponentError.abjad.tools.exceptiontools.MissingComponentError method), 1996
- `__getitem__()` (abjad.tools.exceptiontools.MissingInstrumentError.MissingInstrumentError.abjad.tools.exceptiontools.MissingInstrumentError method), 1997

__getitem__() (abjad.tools.exceptiontools.MissingMarkError.MissingMarkError method), 1998

__getitem__() (abjad.tools.exceptiontools.MissingMeasureError.MissingMeasureError method), 1999

__getitem__() (abjad.tools.exceptiontools.MissingNamedComponentError.MissingNamedComponentError method), 2000

__getitem__() (abjad.tools.exceptiontools.MissingNoteHeadError.MissingNoteHeadError method), 2001

__getitem__() (abjad.tools.exceptiontools.MissingPitchError.MissingPitchError method), 2002

__getitem__() (abjad.tools.exceptiontools.MissingSpannerError.MissingSpannerError method), 2003

__getitem__() (abjad.tools.exceptiontools.MissingTempoError.MissingTempoError method), 2004

__getitem__() (abjad.tools.exceptiontools.MusicContentsError.MusicContentsError method), 2005

__getitem__() (abjad.tools.exceptiontools.NegativeDurationError.NegativeDurationError method), 2006

__getitem__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureError.NonbinaryTimeSignatureError method), 2007

__getitem__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureError.NonbinaryTimeSignatureError method), 2008

__getitem__() (abjad.tools.exceptiontools.NoteHeadError.NoteHeadError method), 2009

__getitem__() (abjad.tools.exceptiontools.OverfullContainerError.OverfullContainerError method), 2010

__getitem__() (abjad.tools.exceptiontools.ParallelError.ParallelError method), 2011

__getitem__() (abjad.tools.exceptiontools.PartitionError.PartitionError method), 2012

__getitem__() (abjad.tools.exceptiontools.PitchError.PitchError method), 2013

__getitem__() (abjad.tools.exceptiontools.SchemeParserFinishedException.SchemeParserFinishedException method), 2014

__getitem__() (abjad.tools.exceptiontools.SpacingError.SpacingError method), 2015

__getitem__() (abjad.tools.exceptiontools.SpannerError.SpannerError method), 2016

__getitem__() (abjad.tools.exceptiontools.SpannerPopulationError.SpannerPopulationError method), 2017

__getitem__() (abjad.tools.exceptiontools.StaffContainmentError.StaffContainmentError method), 2018

__getitem__() (abjad.tools.exceptiontools.TempoError.TempoError method), 2019

__getitem__() (abjad.tools.exceptiontools.TieChainError.TieChainError method), 2020

__getitem__() (abjad.tools.exceptiontools.TimeSignatureAssignmentError.TimeSignatureAssignmentError method), 2021

__getitem__() (abjad.tools.exceptiontools.TimeSignatureError.TimeSignatureError method), 2022

__getitem__() (abjad.tools.exceptiontools.TonalHarmonyError.TonalHarmonyError method), 2023

__getitem__() (abjad.tools.exceptiontools.TupletError.TupletError method), 2024

__getitem__() (abjad.tools.exceptiontools.TupletFuseError.TupletFuseError method), 2025

__getitem__() (abjad.tools.exceptiontools.TypographicWhitespaceError.TypographicWhitespaceError method), 2026

__getitem__() (abjad.tools.exceptiontools.UnboundedTimeIntervalError.UnboundedTimeIntervalError method), 2027

__getitem__() (abjad.tools.exceptiontools.UndefinedSpacingError.UndefinedSpacingError method), 2028

__getitem__() (abjad.tools.exceptiontools.UnderfullContainerError.UnderfullContainerError method), 2029

__getitem__() (abjad.tools.exceptiontools.VoiceContainmentError.VoiceContainmentError method), 2030

__getitem__() (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 481

__getitem__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory method), 660

__getitem__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 864

__getitem__() (abjad.tools.lilypondfiletools.BookPartBlock.BookPartBlock method), 867

__getitem__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile method), 877

__getitem__() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock method), 860

__getitem__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 885

__getitem__() (abjad.tools.lilypondparsertools.SyntaxNode.SyntaxNode.SyntaxNode method), 2078

__getitem__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory method), 938

__getitem__() (abjad.tools.mathtools.Ratio.Ratio.Ratio method), 954

__getitem__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure method), 998

__getitem__() (abjad.tools.measuretools.Measure.Measure.Measure method), 1008

__getitem__() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method), 1051

__getitem__() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn method), 1056

__getitem__() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1058

__getitem__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1136

__getitem__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1138

__getitem__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1153

__getitem__() (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment method), 1088

__getitem__() (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment method), 1096

__getitem__() (abjad.tools.pitchtools.InversionEquivalentChromaticInterval.getitem method), 1160

__getitem__() (abjad.tools.pitchtools.InversionEquivalentChromaticInterval.getitem method), 1410

__getitem__() (abjad.tools.pitchtools.InversionEquivalentChromaticInterval.getitem method), 1419

__getitem__() (abjad.tools.pitchtools.InversionEquivalentDiatonicInterval.getitem method), 1170

__getitem__() (abjad.tools.pitchtools.InversionEquivalentDiatonicInterval.getitem method), 1422

__getitem__() (abjad.tools.pitchtools.InversionEquivalentDiatonicInterval.getitem method), 1431

__getitem__() (abjad.tools.pitchtools.MelodicChromaticInterval.getitem method), 1179

__getitem__() (abjad.tools.pitchtools.MelodicChromaticInterval.getitem method), 1432

__getitem__() (abjad.tools.pitchtools.MelodicChromaticInterval.getitem method), 1435

__getitem__() (abjad.tools.pitchtools.MelodicChromaticInterval.getitem method), 1443

__getitem__() (abjad.tools.pitchtools.MelodicDiatonicInterval.getitem method), 1198

__getitem__() (abjad.tools.pitchtools.MelodicDiatonicInterval.getitem method), 1518

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.getitem method), 1210

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.getitem method), 1524

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchSegment.getitem method), 1216

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchSegment.getitem method), 1533

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchVector.getitem method), 1223

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchVector.getitem method), 1542

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClassCollection.getitem method), 1235

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClassCollection.getitem method), 1505

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.getitem method), 1238

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.getitem method), 1549

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.getitem method), 1245

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.getitem method), 1555

__getitem__() (abjad.tools.pitchtools.ObjectSegment.ObjectSegment.getitem method), 1110

__getitem__() (abjad.tools.pitchtools.ObjectSegment.ObjectSegment.getitem method), 1563

__getitem__() (abjad.tools.pitchtools.ObjectVector.ObjectVector.getitem method), 1115

__getitem__() (abjad.tools.pitchtools.ObjectVector.ObjectVector.getitem method), 1569

__getitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.getitem method), 1252

__getitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.getitem method), 1576

__getitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.getitem method), 1583

__getitem__() (abjad.tools.pitchtools.PitchClassObjectSegment.getitem method), 1119

__getitem__() (abjad.tools.pitchtools.PitchClassObjectSegment.getitem method), 1590

__getitem__() (abjad.tools.pitchtools.PitchObjectSegment.getitem method), 1125

__getitem__() (abjad.tools.pitchtools.PitchObjectSegment.getitem method), 1596

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.getitem method), 1264

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.getitem method), 1603

__getitem__() (abjad.tools.pitchtools.TwelveToneRow.getitem method), 1268

__getitem__() (abjad.tools.pitchtools.TwelveToneRow.getitem method), 1610

__getitem__() (abjad.tools.rhythmtreetools.RhythmTreeContainer.getitem method), 1335

__getitem__() (abjad.tools.rhythmtreetools.RhythmTreeContainer.getitem method), 1511

__getitem__() (abjad.tools.scoretools.GrandStaff.GrandStaff.getitem method), 1376

__getitem__() (abjad.tools.scoretools.GrandStaff.GrandStaff.getitem method), 1617

__getitem__() (abjad.tools.scoretools.PerformerInventory.getitem method), 1383

__getitem__() (abjad.tools.scoretools.PerformerInventory.getitem method), 1623

__getitem__() (abjad.tools.scoretools.PianoStaff.PianoStaff.getitem method), 1392

__getitem__() (abjad.tools.scoretools.PianoStaff.PianoStaff.getitem method), 1630

__getitem__() (abjad.tools.scoretools.Score.Score.Score.getitem method), 1401

__getitem__() (abjad.tools.scoretools.Score.Score.Score.getitem method), 1636

`__getitem__()` (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1661
`__getitem__()` (abjad.tools.stafftools.Staff.Staff.Staff method), 1670
`__getitem__()` (abjad.tools.tietools.TieChain.TieChain.TieChain method), 1683
`__getitem__()` (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner method), 1690
`__getitem__()` (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1709
`__getitem__()` (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method), 1722
`__getitem__()` (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1744
`__getitem__()` (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 1808
`__getitem__()` (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1818
`__getitem__()` (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1838
`__getitem__()` (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1847
`__getitem__()` (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment method), 1860
`__getitem__()` (abjad.tools.voicetools.Voice.Voice.Voice method), 1872
`__getslice__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 382
`__getslice__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 416
`__getslice__()` (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1906
`__getslice__()` (abjad.tools.exceptiontools.AssignabilityError.AssignabilityError.AssignabilityError method), 1974
`__getslice__()` (abjad.tools.exceptiontools.ClefError.ClefError.ClefError method), 1975
`__getslice__()` (abjad.tools.exceptiontools.ContainmentError.ContainmentError.ContainmentError method), 1976
`__getslice__()` (abjad.tools.exceptiontools.ContextContainmentError.ContextContainmentError.ContextContainmentError method), 1977
`__getslice__()` (abjad.tools.exceptiontools.ContiguityError.ContiguityError.ContiguityError method), 1978
`__getslice__()` (abjad.tools.exceptiontools.CyclicNodeError.CyclicNodeError.CyclicNodeError method), 1979
`__getslice__()` (abjad.tools.exceptiontools.DurationError.DurationError.DurationError method), 1980
`__getslice__()` (abjad.tools.exceptiontools.ExtraMarkError.ExtraMarkError.ExtraMarkError method), 1981
`__getslice__()` (abjad.tools.exceptiontools.ExtraNamedComponentError.ExtraNamedComponentError.ExtraNamedComponentError method), 1982
`__getslice__()` (abjad.tools.exceptiontools.ExtraNoteHeadError.ExtraNoteHeadError.ExtraNoteHeadError method), 1983
`__getslice__()` (abjad.tools.exceptiontools.ExtraPitchError.ExtraPitchError.ExtraPitchError method), 1984
`__getslice__()` (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.ExtraSpannerError method), 1985
`__getslice__()` (abjad.tools.exceptiontools.GraceContainerError.GraceContainerError.GraceContainerError method), 1986
`__getslice__()` (abjad.tools.exceptiontools.ImpreciseTempoError.ImpreciseTempoError.ImpreciseTempoError method), 1987
`__getslice__()` (abjad.tools.exceptiontools.InputSpecificationError.InputSpecificationError.InputSpecificationError method), 1988
`__getslice__()` (abjad.tools.exceptiontools.InstrumentError.InstrumentError.InstrumentError method), 1989
`__getslice__()` (abjad.tools.exceptiontools.IntervalError.IntervalError.IntervalError method), 1990
`__getslice__()` (abjad.tools.exceptiontools.IntervalTreeDictionary.IntervalTreeDictionary.IntervalTreeDictionary method), 1991
`__getslice__()` (abjad.tools.exceptiontools.LineBreakError.LineBreakError.LineBreakError method), 1992
`__getslice__()` (abjad.tools.exceptiontools.MarkError.MarkError.MarkError method), 1993
`__getslice__()` (abjad.tools.exceptiontools.MeasureContiguityError.MeasureContiguityError.MeasureContiguityError method), 1994
`__getslice__()` (abjad.tools.exceptiontools.MeasureError.MeasureError.MeasureError method), 1995
`__getslice__()` (abjad.tools.exceptiontools.MissingComponentError.MissingComponentError.MissingComponentError method), 1996
`__getslice__()` (abjad.tools.exceptiontools.MissingInstrumentError.MissingInstrumentError.MissingInstrumentError method), 1997
`__getslice__()` (abjad.tools.exceptiontools.MissingMarkError.MissingMarkError.MissingMarkError method), 1998
`__getslice__()` (abjad.tools.exceptiontools.MissingMeasureError.MissingMeasureError.MissingMeasureError method), 1999
`__getslice__()` (abjad.tools.exceptiontools.MissingNamedComponentError.MissingNamedComponentError.MissingNamedComponentError method), 2000
`__getslice__()` (abjad.tools.exceptiontools.MissingNoteHeadError.MissingNoteHeadError.MissingNoteHeadError method), 2001
`__getslice__()` (abjad.tools.exceptiontools.MissingPitchError.MissingPitchError.MissingPitchError method), 2002
`__getslice__()` (abjad.tools.exceptiontools.MissingSpannerError.MissingSpannerError.MissingSpannerError method), 2003
`__getslice__()` (abjad.tools.exceptiontools.MissingTempoError.MissingTempoError.MissingTempoError method), 2004
`__getslice__()` (abjad.tools.exceptiontools.MusicContentsError.MusicContentsError.MusicContentsError method), 2005
`__getslice__()` (abjad.tools.exceptiontools.NegativeDurationError.NegativeDurationError.NegativeDurationError method), 2006
`__getslice__()` (abjad.tools.exceptiontools.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError method), 2007
`__getslice__()` (abjad.tools.exceptiontools.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError method), 2008
`__getslice__()` (abjad.tools.exceptiontools.NoteHeadError.NoteHeadError.NoteHeadError method), 2009
`__getslice__()` (abjad.tools.exceptiontools.OverfullContainerError.OverfullContainerError.OverfullContainerError method), 2010
`__getslice__()` (abjad.tools.exceptiontools.ParallelError.ParallelError.ParallelError method), 2011

[__getslice__\(\)](#) (abjad.tools.exceptiontools.PartitionError.PartitionError method), 1138
[__getslice__\(\)](#) (abjad.tools.exceptiontools.PitchError.PitchError method), 1153
[__getslice__\(\)](#) (abjad.tools.exceptiontools.SchemeParserFinishedException.SchemeParserFinishedException method), 1088
[__getslice__\(\)](#) (abjad.tools.exceptiontools.SpacingError.SpacingError method), 1096
[__getslice__\(\)](#) (abjad.tools.exceptiontools.SpannerError.SpannerError method), 1160
[__getslice__\(\)](#) (abjad.tools.exceptiontools.SpannerPopulationError.SpannerPopulationError method), 1170
[__getslice__\(\)](#) (abjad.tools.exceptiontools.StaffContainmentError.StaffContainmentError method), 1179
[__getslice__\(\)](#) (abjad.tools.exceptiontools.TempoError.TempoError method), 1184
[__getslice__\(\)](#) (abjad.tools.exceptiontools.TieChainError.TieChainError method), 1198
[__getslice__\(\)](#) (abjad.tools.exceptiontools.TimeSignatureAssignmentError.TimeSignatureAssignmentError method), 1210
[__getslice__\(\)](#) (abjad.tools.exceptiontools.TimeSignatureError.TimeSignatureError method), 1216
[__getslice__\(\)](#) (abjad.tools.exceptiontools.TonalHarmonyError.TonalHarmonyError method), 1238
[__getslice__\(\)](#) (abjad.tools.exceptiontools.TupletError.TupletError method), 1110
[__getslice__\(\)](#) (abjad.tools.exceptiontools.TupletFuseError.TupletFuseError method), 1253
[__getslice__\(\)](#) (abjad.tools.exceptiontools.TypographicWhitespaceError.TypographicWhitespaceError method), 1258
[__getslice__\(\)](#) (abjad.tools.exceptiontools.UnboundedTimeIntervalError.UnboundedTimeIntervalError method), 1119
[__getslice__\(\)](#) (abjad.tools.exceptiontools.UndefinedSpacingError.UndefinedSpacingError method), 1125
[__getslice__\(\)](#) (abjad.tools.exceptiontools.UnderfullContainerError.UnderfullContainerError method), 1264
[__getslice__\(\)](#) (abjad.tools.exceptiontools.VoiceContainmentError.VoiceContainmentError method), 1268
[__getslice__\(\)](#) (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory method), 1384
[__getslice__\(\)](#) (abjad.tools.lilypondfiletools.BookBlock.BookBlock method), 1419
[__getslice__\(\)](#) (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock method), 1432
[__getslice__\(\)](#) (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile method), 1722
[__getslice__\(\)](#) (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock method), 1808
[__getslice__\(\)](#) (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock method), 1818
[__getslice__\(\)](#) (abjad.tools.markuptools.MarkupInventory.MarkupInventory method), 1336
[__getslice__\(\)](#) (abjad.tools.mathtools.Ratio.Ratio method), 954
[__getstate__\(\)](#) (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1138
[__getstate__\(\)](#) (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1153
[__getstate__\(\)](#) (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment method), 1088
[__getstate__\(\)](#) (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSegment.InversionEquivalentChromaticIntervalSegment method), 1096
[__getstate__\(\)](#) (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalSegment.InversionEquivalentDiatonicIntervalSegment method), 1160
[__getstate__\(\)](#) (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment method), 1170
[__getstate__\(\)](#) (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment method), 1179
[__getstate__\(\)](#) (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 1184
[__getstate__\(\)](#) (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1198
[__getstate__\(\)](#) (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment method), 1210
[__getstate__\(\)](#) (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1216
[__getstate__\(\)](#) (abjad.tools.pitchtools.ObjectSegment.ObjectSegment method), 1238
[__getstate__\(\)](#) (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1253
[__getstate__\(\)](#) (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1258
[__getstate__\(\)](#) (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment method), 1119
[__getstate__\(\)](#) (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment method), 1125
[__getstate__\(\)](#) (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory method), 1264
[__getstate__\(\)](#) (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow method), 1268
[__getstate__\(\)](#) (abjad.tools.sequencetools.CyclicList.CyclicList method), 1384
[__getstate__\(\)](#) (abjad.tools.sequencetools.CyclicTuple.CyclicTuple method), 1419
[__getstate__\(\)](#) (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1432
[__getstate__\(\)](#) (abjad.tools.tonalitytools.Scale.Scale method), 1722
[__getstate__\(\)](#) (abjad.tools.tonalitytools.Scale.Scale method), 1808
[__getstate__\(\)](#) (abjad.tools.tonalitytools.Scale.Scale method), 1818
[__getstate__\(\)](#) (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer method), 1336
[__getstate__\(\)](#) (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf method), 1342

__gt__() (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.BassFlute.BassFlute.BassFlute method), 1943

__gt__() (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.BassSaxophone.BassSaxophone.BassSaxophone method), 1945

__gt__() (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.BassTrombone.BassTrombone.BassTrombone method), 1947

__gt__() (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript.BassVoice.BassVoice.BassVoice method), 1950

__gt__() (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.Bassoon.Bassoon.Bassoon method), 1952

__gt__() (abjad.tools.documentationtools.APICrawler.APICrawler.Cello.Cello.Cello method), 1954

__gt__() (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.ClarinetInA.ClarinetInA.ClarinetInA method), 1956

__gt__() (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.Contrabass.Contrabass.Contrabass method), 1957

__gt__() (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ContrabassClarinet.ContrabassClarinet method), 1959

__gt__() (abjad.tools.documentationtools.Documenter.Documenter.ContrabassFlute.ContrabassFlute method), 1961

__gt__() (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.ContrabassSaxophone.ContrabassSaxophone method), 1962

__gt__() (abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.Contrabassoon.Contrabassoon method), 1964

__gt__() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.ContraltoVoice.ContraltoVoice method), 1967

__gt__() (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.EFlatClarinet.EFlatClarinet method), 1969

__gt__() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1971

__gt__() (abjad.tools.durationtools.Duration.Duration method), 447

__gt__() (abjad.tools.durationtools.Offset.Offset method), 451

__gt__() (abjad.tools.durationtools.TimespanConstant.TimespanConstant method), 454

__gt__() (abjad.tools.gracetools.GraceContainer.GraceContainer method), 481

__gt__() (abjad.tools.instrumenttools.Accordion.Accordion method), 490

__gt__() (abjad.tools.instrumenttools.AltoFlute.AltoFlute method), 496

__gt__() (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone method), 502

__gt__() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone method), 508

__gt__() (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet method), 514

__gt__() (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone method), 520

__gt__() (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice method), 526

__gt__() (abjad.tools.instrumenttools.BassClarinet.BassClarinet method), 532

__gt__() (abjad.tools.instrumenttools.BassFlute.BassFlute method), 538

__gt__() (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone method), 544

__gt__() (abjad.tools.instrumenttools.BassTrombone.BassTrombone method), 550

__gt__() (abjad.tools.instrumenttools.BassVoice.BassVoice method), 556

__gt__() (abjad.tools.instrumenttools.Bassoon.Bassoon method), 562

__gt__() (abjad.tools.instrumenttools.Cello.Cello method), 568

__gt__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA method), 574

__gt__() (abjad.tools.instrumenttools.Contrabass.Contrabass method), 580

__gt__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet method), 586

__gt__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute method), 592

__gt__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone method), 598

__gt__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon method), 604

__gt__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice method), 610

__gt__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet method), 616

__gt__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn method), 622

__gt__() (abjad.tools.instrumenttools.Flute.Flute method), 628

__gt__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn method), 633

__gt__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel method), 639

__gt__() (abjad.tools.instrumenttools.Guitar.Guitar method), 645

__gt__() (abjad.tools.instrumenttools.Harp.Harp method), 651

__gt__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord method), 657

__gt__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory method), 660

__gt__() (abjad.tools.instrumenttools.Marimba.Marimba method), 666

__gt__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice method), 672

__gt__() (abjad.tools.instrumenttools.Oboe.Oboe method), 678

__gt__() (abjad.tools.instrumenttools.Piano.Piano method), 684

__gt__() (abjad.tools.instrumenttools.Piccolo.Piccolo method), 690

`__gt__()` (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone.LilyPondFraction.LilyPondFraction method), 696
`__gt__()` (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone.LilyPondLexicalDefinition.LilyPondLexicalDefinition method), 702
`__gt__()` (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.LilyPondParsertools.LilyPondParser.LilyPondParser.LilyPondParser method), 708
`__gt__()` (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.LilyPondParsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 714
`__gt__()` (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.LilyPondParsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser method), 720
`__gt__()` (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice.LilyPondParsertools.SchemeParser.SchemeParser.SchemeParser method), 726
`__gt__()` (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet.LilyPondParsertools.SyntaxNode.SyntaxNode.SyntaxNode method), 732
`__gt__()` (abjad.tools.instrumenttools.Tuba.Tuba.Tuba.LilyPondParsertools.Annotation.Annotation.Annotation method), 738
`__gt__()` (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.LilyPondParsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser method), 744
`__gt__()` (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone.LilyPondParsertools.Annotation.Annotation.Annotation method), 750
`__gt__()` (abjad.tools.instrumenttools.Viola.Viola.Viola.LilyPondParsertools.Annotation.Annotation.Annotation method), 756
`__gt__()` (abjad.tools.instrumenttools.Violin.Violin.Violin.LilyPondParsertools.Annotation.Annotation.Annotation method), 762
`__gt__()` (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone.LilyPondParsertools.Annotation.Annotation.Annotation method), 768
`__gt__()` (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication.LilyPondComment.LilyPondComment.LilyPondComment method), 824
`__gt__()` (abjad.tools.leaftools.Leaf.Leaf.Leaf.LilyPondComment.LilyPondComment.LilyPondComment method), 830
`__gt__()` (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken.LilyPondComment.LilyPondComment.LilyPondComment method), 862
`__gt__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock.LilyPondComment.LilyPondComment.LilyPondComment method), 865
`__gt__()` (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock.LilyPondComment.LilyPondComment.LilyPondComment method), 868
`__gt__()` (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken.LilyPondComment.LilyPondComment.LilyPondComment method), 871
`__gt__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile.LilyPondComment.LilyPondComment.LilyPondComment method), 877
`__gt__()` (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondComment.LilyPondComment.LilyPondComment method), 879
`__gt__()` (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken.LilyPondComment.LilyPondComment.LilyPondComment method), 880
`__gt__()` (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock.LilyPondComment.LilyPondComment.LilyPondComment method), 860
`__gt__()` (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock.LilyPondComment.LilyPondComment.LilyPondComment method), 885
`__gt__()` (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy.LilyPondComment.LilyPondComment.LilyPondComment method), 2036
`__gt__()` (abjad.tools.lilypondparsertools.LilyPondDuration.LilyPondDuration.LilyPondDuration.LilyPondComment.LilyPondComment.LilyPondComment method), 2037
`__gt__()` (abjad.tools.lilypondparsertools.LilyPondEvent.LilyPondEvent.LilyPondEvent.LilyPondComment.LilyPondComment.LilyPondComment method), 2038

gt() (abjad.tools.noteheadtools.NoteHead.NoteHead.NoteHead method), 1043	_gt_() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1150
gt() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method), 1051	_gt_() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1153
gt() (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell method), 1054	_gt_() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1156
gt() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1056	_gt_() (abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject method), 1083
gt() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1058	_gt_() (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject method), 1085
gt() (abjad.tools.pitchtools.Accidental.Accidental.Accidental method), 1130	_gt_() (abjad.tools.pitchtools.HarmonicObject.HarmonicObject.HarmonicObject method), 1086
gt() (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject.ChromaticIntervalClassObject method), 1065	_gt_() (abjad.tools.pitchtools.IntervalClassObjectKeySet.IntervalClassObjectKeySet method), 1088
gt() (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.ChromaticIntervalObject method), 1066	_gt_() (abjad.tools.pitchtools.IntervalObjectSet.IntervalClassObjectKeySet method), 1090
gt() (abjad.tools.pitchtools.ChromaticObject.ChromaticObject.ChromaticObject method), 1068	_gt_() (abjad.tools.pitchtools.IntervalObject.IntervalObject.IntervalObject method), 1092
gt() (abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject.ChromaticPitchObject method), 1069	_gt_() (abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass.IntervalObjectClass method), 1094
gt() (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject.CounterpointIntervalClassObject method), 1071	_gt_() (abjad.tools.pitchtools.IntervalObjectSegment.IntervalClassObjectKeySet method), 1096
gt() (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.CounterpointIntervalObject method), 1072	_gt_() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet method), 1098
gt() (abjad.tools.pitchtools.CounterpointObject.CounterpointObject.CounterpointObject method), 1074	_gt_() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassObject.InversionEquivalentChromaticIntervalClassObject method), 1158
gt() (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject.DiatonicIntervalClassObject method), 1075	_gt_() (abjad.tools.pitchtools.IntervalInEquivalClassObject.IntervalInEquivalClassObject method), 1160
gt() (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.DiatonicIntervalObject method), 1077	_gt_() (abjad.tools.pitchtools.IntervalEquivalentChromaticIntervalClassObject.IntervalEquivalentChromaticIntervalClassObject method), 1163
gt() (abjad.tools.pitchtools.DiatonicObject.DiatonicObject.DiatonicObject method), 1078	_gt_() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassObject.InversionEquivalentChromaticIntervalClassObject method), 1166
gt() (abjad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClassObject.DiatonicPitchClassObject method), 1080	_gt_() (abjad.tools.pitchtools.IntervalEquivalentDiatonicIntervalClassObject.IntervalEquivalentDiatonicIntervalClassObject method), 1168
gt() (abjad.tools.pitchtools.DiatonicPitchObject.DiatonicPitchObject.DiatonicPitchObject method), 1081	_gt_() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassObject.InversionEquivalentDiatonicIntervalClassObject method), 1170
gt() (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval method), 1132	_gt_() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassObject.InversionEquivalentDiatonicIntervalClassObject method), 1173
gt() (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 1133	_gt_() (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1175
gt() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1136	_gt_() (abjad.tools.pitchtools.MelodicClassVector.HarmonicChromaticIntervalClassVector method), 1177
gt() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1138	_gt_() (abjad.tools.pitchtools.MelodicClassVector.HarmonicChromaticIntervalClassVector method), 1179
gt() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1141	_gt_() (abjad.tools.pitchtools.MelodicClassVector.HarmonicChromaticIntervalClassVector method), 1182
gt() (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 1143	_gt_() (abjad.tools.pitchtools.MelodicClassVector.HarmonicChromaticIntervalClassVector method), 1184
gt() (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1144	_gt_() (abjad.tools.pitchtools.MelodicClassVector.HarmonicChromaticIntervalClassVector method), 1188
gt() (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval method), 1146	_gt_() (abjad.tools.pitchtools.MelodicClassVector.HarmonicChromaticIntervalClassVector method), 1190
gt() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1148	_gt_() (abjad.tools.pitchtools.MelodicClassVector.HarmonicChromaticIntervalClassVector method), 1192

- __gt__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicIntervalClassObjectSegment.ObjectSegment.ObjectSegment method), 1194
- __gt__() (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicIntervalClassObjectSet.ObjectSet.ObjectSet method), 1196
- __gt__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicIntervalSegmentObjectSet.ObjectSet.ObjectSet method), 1198
- __gt__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicIntervalSetObjectSet.ObjectSet.ObjectSet method), 1201
- __gt__() (abjad.tools.pitchtools.MelodicIntervalClassObject.MelodicIntervalClassObjectSet.ObjectSet.ObjectSet method), 1100
- __gt__() (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObjectSet.ObjectSet.ObjectSet method), 1102
- __gt__() (abjad.tools.pitchtools.MelodicObject.MelodicObjectSet.ObjectSet.ObjectSet method), 1103
- __gt__() (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitchClassObjectSegment.PitchClassObjectSegment.PitchClassObjectSegment method), 1206
- __gt__() (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1208
- __gt__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegmentObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1210
- __gt__() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSetObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1213
- __gt__() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegmentObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1217
- __gt__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSetObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1220
- __gt__() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVectorObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1223
- __gt__() (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1227
- __gt__() (abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1229
- __gt__() (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1231
- __gt__() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1233
- __gt__() (abjad.tools.pitchtools.NumberedChromaticPitchClassColor.NamedChromaticPitchClassColorObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1235
- __gt__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NamedChromaticPitchClassSegmentObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1238
- __gt__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NamedChromaticPitchClassSetObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1242
- __gt__() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NamedChromaticPitchClassVectorObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1245
- __gt__() (abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1248
- __gt__() (abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1250
- __gt__() (abjad.tools.pitchtools.NumberedObject.NumberedObjectSet.ObjectSet.ObjectSet method), 1105
- __gt__() (abjad.tools.pitchtools.NumberedPitchClassObject.NumberedPitchClassObjectSet.ObjectSet.ObjectSet method), 1106
- __gt__() (abjad.tools.pitchtools.NumberedPitchObject.NumberedPitchObjectSet.ObjectSet.ObjectSet method), 1108

- `__gt__()` (abjad.tools.timetoken.tools.NoteFilledTimeTokenMaker.NoFilledTimeTokenMaker.NoFilledTimeTokenMaker.Moment.Verti method), 1772
- `__gt__()` (abjad.tools.timetoken.tools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker.Ou method), 1776
- `__gt__()` (abjad.tools.timetoken.tools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker.NoteOutputBeas method), 1779
- `__gt__()` (abjad.tools.timetoken.tools.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker.OutputIncise method), 1782
- `__gt__()` (abjad.tools.timetoken.tools.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker.Dis method), 1763
- `__gt__()` (abjad.tools.timetoken.tools.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.DuplicateIdCl method), 1784
- `__gt__()` (abjad.tools.timetoken.tools.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.EmptyCo method), 1788
- `__gt__()` (abjad.tools.timetoken.tools.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker.Check.Intern method), 1790
- `__gt__()` (abjad.tools.timetoken.tools.TimeTokenMaker.TimeTokenMaker.TimeTokenMaker.Wellformednesstools.MisdurationMeasureCheck.Misdu method), 1765
- `__gt__()` (abjad.tools.timetoken.tools.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurn method), 1793
- `__gt__()` (abjad.tools.timetoken.tools.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker.TokenIncised method), 1796
- `__gt__()` (abjad.tools.timetoken.tools.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker.TokenIncised method), 1799
- `__gt__()` (abjad.tools.timetoken.tools.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.MissingPar method), 1767
- `__gt__()` (abjad.tools.timetoken.tools.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker.NestedMe method), 1802
- `__gt__()` (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.abjad.tools.wellformednesstools.OverlappingBeamCheck.Overlap method), 1805
- `__gt__()` (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator.OverlappingGlissandoCheck.Ove method), 1808
- `__gt__()` (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator.abjad.tools.wellformednesstools.OverlappingOctavationCheck.Ov method), 1810
- `__gt__()` (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator.abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpin method), 1811
- `__gt__()` (abjad.tools.tonalitytools.Mode.Mode.Mode __hash__()) (abjad.tools.contexttools.InstrumentMark.InstrumentMark.Instru method), 1812
- `__gt__()` (abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator.abjad.tools.durationtools.Duration.Duration.Duration method), 1813
- `__gt__()` (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.QualityIndicator.abjad.tools.offsettools.Offset.Offset.Offset method), 1815
- `__gt__()` (abjad.tools.tonalitytools.Scale.Scale.Scale __hash__()) (abjad.tools.instrumenttools.Accordion.Accordion.Accordion method), 1818
- `__gt__()` (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree.abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method), 1819
- `__gt__()` (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator.abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.Alt method), 1821
- `__gt__()` (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction.abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.Alto method), 1822
- `__gt__()` (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.abjad.tools.tuplettools.BFlatClarinet.BFlatClarinet.BFlatC method), 1838
- `__gt__()` (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet __hash__()) (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxop method), 1847

__hash__() (abjad.tools.instrumenttools.BartitoneVoice.BartitoneVoice.BartitoneVoice.__hash__ method), 526

__hash__() (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet.__hash__ method), 532

__hash__() (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute.__hash__ method), 538

__hash__() (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone.__hash__ method), 545

__hash__() (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone.__hash__ method), 550

__hash__() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice.__hash__ method), 556

__hash__() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon.__hash__ method), 562

__hash__() (abjad.tools.instrumenttools.Cello.Cello.Cello.__hash__ method), 568

__hash__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA.__hash__ method), 574

__hash__() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass.__hash__ method), 580

__hash__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet.__hash__ method), 586

__hash__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute.__hash__ method), 592

__hash__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone.__hash__ method), 599

__hash__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon.__hash__ method), 604

__hash__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice.__hash__ method), 610

__hash__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet.__hash__ method), 616

__hash__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn.__hash__ method), 622

__hash__() (abjad.tools.instrumenttools.Flute.Flute.Flute.__hash__ method), 628

__hash__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn.__hash__ method), 634

__hash__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel.__hash__ method), 639

__hash__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar.__hash__ method), 645

__hash__() (abjad.tools.instrumenttools.Harp.Harp.Harp.__hash__ method), 651

__hash__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord.__hash__ method), 657

__hash__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba.__hash__ method), 666

__hash__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice.__hash__ method), 672

__hash__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe.__hash__ method), 678

__hash__() (abjad.tools.instrumenttools.Piano.Piano.Piano.__hash__ method), 684

__hash__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo.__hash__ method), 690

__hash__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone.__hash__ method), 697

__hash__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.__hash__ method), 703

__hash__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.__hash__ method), 708

__hash__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.__hash__ method), 715

__hash__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice.__hash__ method), 720

__hash__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet.__hash__ method), 726

__hash__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba.__hash__ method), 732

__hash__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.__hash__ method), 738

__hash__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone.__hash__ method), 744

__hash__() (abjad.tools.instrumenttools.Viola.Viola.Viola.__hash__ method), 750

__hash__() (abjad.tools.instrumenttools.Violin.Violin.Violin.__hash__ method), 756

__hash__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone.__hash__ method), 762

__hash__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.__hash__ method), 768

__hash__() (abjad.tools.mathtools.Ratio.Ratio.Ratio.__hash__ method), 774

__hash__() (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject.__hash__ method), 1065

__hash__() (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.__hash__ method), 1067

__hash__() (abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject.__hash__ method), 1070

__hash__() (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject.__hash__ method), 1071

__hash__() (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.__hash__ method), 1073

__hash__() (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject.__hash__ method), 1075

__hash__() (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.__hash__ method), 1077

__hash__() (abjad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClassObject.__hash__ method), 1080

__hash__() (abjad.tools.pitchtools.DiatonicPitchObject.DiatonicPitchObject.__hash__ method), 1082

__hash__() (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.__hash__ method), 1132

__hash__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassSet.Hashon(), abjad.tools.pitchtools.MelodicChromaticIntervalClassSet.Hashon(), 1133
method), 1133

__hash__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.Hashon(), abjad.tools.pitchtools.MelodicChromaticIntervalSegment.Hashon(), 1138
method), 1138

__hash__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.Hashon(), abjad.tools.pitchtools.MelodicChromaticIntervalSet.Hashon(), 1141
method), 1141

__hash__() (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.Hashon(), abjad.tools.pitchtools.MelodicCounterpointIntervalClass.Hashon(), 1143
method), 1143

__hash__() (abjad.tools.pitchtools.HarmonicCounterpointIntervalClassSet.Hashon(), abjad.tools.pitchtools.MelodicCounterpointIntervalClassSet.Hashon(), 1144
method), 1144

__hash__() (abjad.tools.pitchtools.HarmonicDiatonicInterval.Hashon(), abjad.tools.pitchtools.MelodicDiatonicInterval.Hashon(), 1146
method), 1146

__hash__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.Hashon(), abjad.tools.pitchtools.MelodicDiatonicIntervalClass.Hashon(), 1148
method), 1148

__hash__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.Hashon(), abjad.tools.pitchtools.MelodicDiatonicIntervalClassSet.Hashon(), 1151
method), 1151

__hash__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.Hashon(), abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.Hashon(), 1153
method), 1153

__hash__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.Hashon(), abjad.tools.pitchtools.MelodicDiatonicIntervalSet.Hashon(), 1156
method), 1156

__hash__() (abjad.tools.pitchtools.HarmonicIntervalClassObject.Hashon(), abjad.tools.pitchtools.MelodicIntervalClassObject.Hashon(), 1083
method), 1083

__hash__() (abjad.tools.pitchtools.HarmonicIntervalObject.Hashon(), abjad.tools.pitchtools.MelodicIntervalObject.Hashon(), 1085
method), 1085

__hash__() (abjad.tools.pitchtools.IntervalClassObjectSegment.Hashon(), abjad.tools.pitchtools.MelodicIntervalClassObjectSegment.Hashon(), 1088
method), 1088

__hash__() (abjad.tools.pitchtools.IntervalClassObjectSet.Hashon(), abjad.tools.pitchtools.MelodicIntervalClassObjectSet.Hashon(), 1090
method), 1090

__hash__() (abjad.tools.pitchtools.IntervalObject.Hashon(), abjad.tools.pitchtools.MelodicIntervalObject.Hashon(), 1092
method), 1092

__hash__() (abjad.tools.pitchtools.IntervalObjectClass.Hashon(), abjad.tools.pitchtools.MelodicIntervalObjectClass.Hashon(), 1094
method), 1094

__hash__() (abjad.tools.pitchtools.IntervalObjectSegment.Hashon(), abjad.tools.pitchtools.MelodicIntervalObjectSegment.Hashon(), 1096
method), 1096

__hash__() (abjad.tools.pitchtools.IntervalObjectSet.Hashon(), abjad.tools.pitchtools.MelodicIntervalObjectSet.Hashon(), 1098
method), 1098

__hash__() (abjad.tools.pitchtools.InversionEquivalentChromaticInterval.Hashon(), abjad.tools.pitchtools.MelodicInversionEquivalentChromaticInterval.Hashon(), 1158
method), 1158

__hash__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSegment.Hashon(), abjad.tools.pitchtools.MelodicInversionEquivalentChromaticIntervalSegment.Hashon(), 1160
method), 1160

__hash__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSet.Hashon(), abjad.tools.pitchtools.MelodicInversionEquivalentChromaticIntervalSet.Hashon(), 1163
method), 1163

__hash__() (abjad.tools.pitchtools.InversionEquivalentDiatonicInterval.Hashon(), abjad.tools.pitchtools.MelodicInversionEquivalentDiatonicInterval.Hashon(), 1168
method), 1168

__hash__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalSegment.Hashon(), abjad.tools.pitchtools.MelodicInversionEquivalentDiatonicIntervalSegment.Hashon(), 1170
method), 1170

__hash__() (abjad.tools.pitchtools.MelodicChromaticInterval.Hashon(), abjad.tools.pitchtools.MelodicChromaticInterval.Hashon(), 1175
method), 1175

__hash__() (abjad.tools.pitchtools.MelodicChromaticIntervalClass.Hashon(), abjad.tools.pitchtools.MelodicChromaticIntervalClass.Hashon(), 1177
method), 1177

__hash__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSet.Hashon(), abjad.tools.pitchtools.MelodicChromaticIntervalClassSet.Hashon(), 1179
method), 1179

__hash__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.Hashon(), abjad.tools.pitchtools.MelodicChromaticIntervalSegment.Hashon(), 1185
method), 1185

[__hash__\(\)](#) (abjad.tools.pitchtools.PitchClassObject.PitchClassObject.PitchClassObject), 1117
[__hash__\(\)](#) (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment.PitchClassObjectSegment), 1119
[__hash__\(\)](#) (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet), 1121
[__hash__\(\)](#) (abjad.tools.pitchtools.PitchObject.PitchObject.PitchObject), 1124
[__hash__\(\)](#) (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment.PitchObjectSegment), 1125
[__hash__\(\)](#) (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet), 1128
[__hash__\(\)](#) (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow), 1268
[__hash__\(\)](#) (abjad.tools.scoretools.Performer.Performer.Performer), 1381
[__hash__\(\)](#) (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple), 1433
[__hash__\(\)](#) (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval), 1709
[__hash__\(\)](#) (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass), 1805
[__hash__\(\)](#) (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator), 1808
[__hash__\(\)](#) (abjad.tools.tonalitytools.Scale.Scale.Scale), 1818
[__hash__\(\)](#) (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment), 1860
[__iadd__\(\)](#) (abjad.tools.containertools.Cluster.Cluster.Cluster), 341
[__iadd__\(\)](#) (abjad.tools.containertools.Container.Container.Container), 347
[__iadd__\(\)](#) (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer), 355
[__iadd__\(\)](#) (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory), 382
[__iadd__\(\)](#) (abjad.tools.contexttools.Context.Context.Context), 391
[__iadd__\(\)](#) (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory), 416
[__iadd__\(\)](#) (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory), 1907
[__iadd__\(\)](#) (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer), 481
[__iadd__\(\)](#) (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory), 660
[__iadd__\(\)](#) (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock), 865
[__iadd__\(\)](#) (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock), 868
[__iadd__\(\)](#) (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile), 877
[__iadd__\(\)](#) (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock), 860
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock), 885
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.MarkupInventory.MarkupInventory.MarkupInventory), 939
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure), 987
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.DynamicMeasure.DynamicMeasure.DynamicMeasure), 999
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.Measure.Measure.Measure), 1008
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.PitchArray.PitchArray.PitchArray), 1051
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.PitchArrayRow.PitchArrayRow.PitchArrayRow), 1058
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping), 1253
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory), 1258
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory), 1264
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.GrandStaff.GrandStaff.GrandStaff), 1376
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.PerformerInventory.PerformerInventory.PerformerInventory), 1384
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.PianoStaff.PianoStaff.PianoStaff), 1392
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.Score.Score.Score), 1401
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.StaffGroup.StaffGroup.StaffGroup), 1410
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.CyclicList.CyclicList.CyclicList), 1419
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.RhythmicStaff.RhythmicStaff.RhythmicStaff), 1661
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.Staff.Staff.Staff), 1670
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet), 1838
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.Tuplet.Tuplet.Tuplet), 1847
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.Voice.Voice.Voice), 1872
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.Cluster.Cluster.Cluster), 341
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.Container.Container.Container), 347
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer), 355
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory), 382
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.Context.Context.Context), 391
[__iadd__\(\)](#) (abjad.tools.lylilypondfiletools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory), 416

__imul__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ChromaticIntervalObject.ChromaticIntervalObject method), 1907

__imul__() (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer.ChromaticPitchObject.ChromaticPitchObject method), 481

__imul__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory.CounterpointIntervalClassObject.CounterpointIntervalClassObject method), 660

__imul__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject method), 865

__imul__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock.DiatonicIntervalClassObject.DiatonicIntervalClassObject method), 868

__imul__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject method), 877

__imul__() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock.DiatonicPitchClassObject.DiatonicPitchClassObject method), 860

__imul__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock.pitchtools.DiatonicPitchObject.DiatonicPitchObject method), 885

__imul__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory.HarmonicChromaticInterval.HarmonicChromaticInterval method), 939

__imul__() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 987

__imul__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 999

__imul__() (abjad.tools.measuretools.Measure.Measure.Measure.__int__() (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1008

__imul__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1253

__imul__() (abjad.tools.pitchtools.OctaveTranspositionMappingInvert.OctaveTranspositionMappingInvert.OctaveTranspositionMappingInvert.OctaveTranspositionMappingInvert method), 1258

__imul__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory.HarmonicIntervalClassObject.HarmonicIntervalClassObject method), 1264

__imul__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject method), 1376

__imul__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory.IntervalObject.IntervalObject.IntervalObject method), 1384

__imul__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff (abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass method), 1393

__imul__() (abjad.tools.scoretools.Score.Score.Score __int__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1401

__imul__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass method), 1410

__imul__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval method), 1419

__imul__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1661

__imul__() (abjad.tools.stafftools.Staff.Staff.Staff __int__() (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval method), 1670

__imul__() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass method), 1838

__imul__() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet __int__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1847

__imul__() (abjad.tools.voicetools.Voice.Voice.Voice __int__() (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass method), 1872

__int__() (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject.ChromaticIntervalClassObject.ChromaticIntervalClassObject method), 1065

- `__int__()` (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject.MelodicIntervalObject.ChromaticIntervalSegment.HarmonicIntervalSegment method), 1102
- `__int__()` (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch.ChromaticIntervalSet.HarmonicIntervalSet method), 1206
- `__int__()` (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.NamedChromaticPitchClass.ClassSet.HarmonicIntervalSet method), 1208
- `__int__()` (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch.HarmonicDiatonicIntervalSegment.HarmonicIntervalSegment method), 1227
- `__int__()` (abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass.NamedDiatonicPitchClass.IntervalSet.HarmonicDiatonicIntervalSet method), 1229
- `__int__()` (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch.NumberedChromaticPitch.IntervalClassSegment.IntervalClassSegment method), 1231
- `__int__()` (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass.NumberedChromaticPitchClass.ObjectSet.IntervalClassSegment method), 1233
- `__int__()` (abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch.NumberedDiatonicPitch.IntervalObjectSegment.IntervalObjectSegment method), 1248
- `__int__()` (abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass.ObjectSet.IntervalObjectSegment method), 1250
- `__int__()` (abjad.tools.pitchtools.NumberedPitchObject.NumberedPitchObject.NumberedPitchObject.IntervalEquivalentChromaticIntervalClassSegment.IntervalEquivalentChromaticIntervalClassSegment method), 1108
- `__int__()` (abjad.tools.pitchtools.PitchObject.PitchObject.PitchObject) (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.IntervalEquivalentChromaticIntervalClassSegment method), 1124
- `__iter__()` (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig.abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.IntervalEquivalentChromaticIntervalClassSegment method), 1897
- `__iter__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.InversionEquivalentDiatonicIntervalClassSegment.IntervalEquivalentDiatonicIntervalClassSegment method), 382
- `__iter__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.IntervalEquivalentDiatonicIntervalClassSegment.IntervalEquivalentDiatonicIntervalClassSegment method), 416
- `__iter__()` (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary.abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment method), 1904
- `__iter__()` (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 1907
- `__iter__()` (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph.abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment method), 1967
- `__iter__()` (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler.abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1969
- `__iter__()` (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory.abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 660
- `__iter__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock.abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 865
- `__iter__()` (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock.abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 868
- `__iter__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile.abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 877
- `__iter__()` (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock.ChromaticPitchSegment.NamedChromaticPitchSegment method), 860
- `__iter__()` (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock.abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet method), 885
- `__iter__()` (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory.abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector method), 939
- `__iter__()` (abjad.tools.mathtools.Ratio.Ratio.Ratio) (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 955
- `__iter__()` (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.ChromaticIntervalClassVector method), 1136

- [__le__\(\) \(abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.MahJazKeySignatureMark.tools.APICrawler.APICrawler.APICrawler method\), 405](#)
- [__le__\(\) \(abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark.tools.AbjadAPIGenerator.AbjadAPIGenerator method\), 409](#)
- [__le__\(\) \(abjad.tools.contexttools.TempoMark.TempoMark.TempoMark.abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler method\), 414](#)
- [__le__\(\) \(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter method\), 416](#)
- [__le__\(\) \(abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark.abjad.tools.documentationtools.Documenter.Documenter.Documenter method\), 421](#)
- [__le__\(\) \(abjad.tools.datastructuretools.Digraph.Digraph.Digraph.abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler method\), 1902](#)
- [__le__\(\) \(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary.abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter method\), 1904](#)
- [__le__\(\) \(abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method\), 1907](#)
- [__le__\(\) \(abjad.tools.datastructuretools.OrdinalConstant.OrdinalConstant.OrdinalConstant.abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler method\), 1909](#)
- [__le__\(\) \(abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript.abjad.tools.documentationtools.Pipe.Pipe.Pipe method\), 1916](#)
- [__le__\(\) \(abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript.abjad.tools.documentationtools.Duration.Duration.Duration method\), 1919](#)
- [__le__\(\) \(abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript.abjad.tools.documentationtools.Offset.Offset.Offset method\), 1921](#)
- [__le__\(\) \(abjad.tools.developerscripttools.CleanScript.CleanScript.CleanScript.abjad.tools.durationtools.TimespanConstant.TimespanConstant.TimespanConstant method\), 1923](#)
- [__le__\(\) \(abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScript.abjad.tools.durationtools.GraceContainer.GraceContainer.GraceContainer method\), 1926](#)
- [__le__\(\) \(abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScript.abjad.tools.durationtools.Accordion.Accordion.Accordion method\), 1928](#)
- [__le__\(\) \(abjad.tools.developerscripttools.DeveloperScript.DeveloperScript.DeveloperScript.abjad.tools.durationtools.AltoFlute.AltoFlute.AltoFlute method\), 1911](#)
- [__le__\(\) \(abjad.tools.developerscripttools.DirectoryScript.DirectoryScript.DirectoryScript.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method\), 1913](#)
- [__le__\(\) \(abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript.abjad.tools.durationtools.MakeNewClassTemplateScript method\), 1930](#)
- [__le__\(\) \(abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.abjad.tools.durationtools.MakeNewFunctionTemplateScript method\), 1933](#)
- [__le__\(\) \(abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript.RenameModulesScript.abjad.tools.durationtools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone method\), 1935](#)
- [__le__\(\) \(abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript.abjad.tools.durationtools.BassVoice.BassVoice.BassVoice method\), 1938](#)
- [__le__\(\) \(abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript.ReplacePromptsScript.abjad.tools.durationtools.BassClarinet.BassClarinet.BassClarinet method\), 1941](#)
- [__le__\(\) \(abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScript.abjad.tools.durationtools.BassFlute.BassFlute.BassFlute method\), 1943](#)
- [__le__\(\) \(abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript.abjad.tools.durationtools.BassSaxophone.BassSaxophone.BassSaxophone method\), 1945](#)
- [__le__\(\) \(abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript.abjad.tools.durationtools.BassTrombone.BassTrombone.BassTrombone method\), 1948](#)
- [__le__\(\) \(abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript.SvnMessageScript.abjad.tools.durationtools.BassVoice.BassVoice.BassVoice method\), 1950](#)
- [__le__\(\) \(abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript.abjad.tools.durationtools.Bassoon.Bassoon.Bassoon method\), 1952](#)

- `__le__()` (abjad.tools.instrumenttools.Cello.Cello.Cello method), 568
- `__le__()` (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 575
- `__le__()` (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 580
- `__le__()` (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 587
- `__le__()` (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 593
- `__le__()` (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 599
- `__le__()` (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 605
- `__le__()` (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 610
- `__le__()` (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 617
- `__le__()` (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 623
- `__le__()` (abjad.tools.instrumenttools.Flute.Flute.Flute method), 628
- `__le__()` (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 634
- `__le__()` (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 639
- `__le__()` (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 645
- `__le__()` (abjad.tools.instrumenttools.Harp.Harp.Harp method), 652
- `__le__()` (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 658
- `__le__()` (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 660
- `__le__()` (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 666
- `__le__()` (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 672
- `__le__()` (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 679
- `__le__()` (abjad.tools.instrumenttools.Piano.Piano.Piano method), 685
- `__le__()` (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 691
- `__le__()` (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 697
- `__le__()` (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 703
- `__le__()` (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 708
- `__le__()` (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 715
- `__le__()` (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 721
- `__le__()` (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 726
- `__le__()` (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 732
- `__le__()` (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 738
- `__le__()` (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 744
- `__le__()` (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 750
- `__le__()` (abjad.tools.instrumenttools.Viola.Viola.Viola method), 756
- `__le__()` (abjad.tools.instrumenttools.Violin.Violin.Violin method), 762
- `__le__()` (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 768
- `__le__()` (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 824
- `__le__()` (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 830
- `__le__()` (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 862
- `__le__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 865
- `__le__()` (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 868
- `__le__()` (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken method), 871
- `__le__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 877
- `__le__()` (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 879
- `__le__()` (abjad.tools.lilypondfiletools.LilyPondHeaderToken.LilyPondHeaderToken.LilyPondHeaderToken method), 880
- `__le__()` (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 860
- `__le__()` (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 885
- `__le__()` (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy method), 2036
- `__le__()` (abjad.tools.lilypondparsertools.LilyPondDuration.LilyPondDuration.LilyPondDuration method), 2037
- `__le__()` (abjad.tools.lilypondparsertools.LilyPondEvent.LilyPondEvent.LilyPondEvent method), 2038
- `__le__()` (abjad.tools.lilypondparsertools.LilyPondFraction.LilyPondFraction.LilyPondFraction method), 2040
- `__le__()` (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition.LilyPondLexicalDefinition method), 2043
- `__le__()` (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser method), 2047
- `__le__()` (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2067
- `__le__()` (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser method), 2073

__le__() (abjad.tools.lilypondparsertools.SchemeParser.SchemeParserSchemeParserPitchtools.Accidental.Accidental.Accidental method), 2077

__le__() (abjad.tools.lilypondparsertools.SyntaxNode.SyntaxNodeSyntaxNodeSyntaxtools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject.ChromaticIntervalClassObject method), 2079

__le__() (abjad.tools.marktools.Annotation.Annotation.Annotation) (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.ChromaticIntervalObject method), 892

__le__() (abjad.tools.marktools.Articulation.Articulation.Articulation) (abjad.tools.pitchtools.ChromaticObject.ChromaticObject.ChromaticObject method), 895

__le__() (abjad.tools.marktools.BarLine.BarLine.BarLine) (abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject.ChromaticPitchObject method), 899

__le__() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter) (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject.CounterpointIntervalClassObject method), 902

__le__() (abjad.tools.marktoolsDirectedMarkDirectedMarkDirectedMark) (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.CounterpointIntervalObject method), 889

__le__() (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMarkLilyPondCommandMarkLilyPondCommandMark) (abjad.tools.pitchtools.CounterpointObject.CounterpointObject.CounterpointObject method), 905

__le__() (abjad.tools.marktools.LilyPondComment.LilyPondCommentLilyPondCommentLilyPondComment) (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject.DiatonicIntervalClassObject method), 908

__le__() (abjad.tools.marktools.Mark.Mark.Mark) (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.DiatonicIntervalObject method), 910

__le__() (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo) (abjad.tools.pitchtools.DiatonicObject.DiatonicObject.DiatonicObject method), 913

__le__() (abjad.tools.markuptools.Markup.Markup.Markup) (abjad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClassObject.DiatonicPitchClassObject method), 934

__le__() (abjad.tools.markuptools.MarkupCommand.MarkupCommandMarkupCommandMarkupCommand) (abjad.tools.pitchtools.DiatonicPitchObject.DiatonicPitchObject.DiatonicPitchObject method), 936

__le__() (abjad.tools.markuptools.MarkupInventory.MarkupInventoryMarkupInventoryMarkupInventory) (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval method), 939

__le__() (abjad.tools.mathtools.BoundedObject.BoundedObjectBoundedObjectBoundedObject) (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 946

__le__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFractionNonreducedFractionNonreducedFraction) (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 950

__le__() (abjad.tools.mathtools.Ratio.Ratio.Ratio) (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 955

__le__() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasureAnonymousMeasureAnonymousMeasure) (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 987

__le__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasureDynamicMeasureDynamicMeasure) (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 999

__le__() (abjad.tools.measuretools.Measure.Measure.Measure) (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1009

__le__() (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic) (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval method), 1035

__le__() (abjad.tools.notetools.Note.Note.Note) (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1040

__le__() (abjad.tools.notetools.NoteHead.NoteHead.NoteHead) (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1043

__le__() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray) (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1051

__le__() (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCellPitchArrayCellPitchArrayCell) (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1054

__le__() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumnPitchArrayColumnPitchArrayColumn) (abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject.HarmonicIntervalClassObject method), 1056

__le__() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRowPitchArrayRowPitchArrayRow) (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.HarmonicIntervalObject method), 1058

- `__le__()` (abjad.tools.pitchtools.HarmonicObject.HarmonicObject.HarmonicObject.pitchtools.MelodicIntervalObject.MelodicIntervalObject method), 1086
- `__le__()` (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment.IntervalClassObjectSegment.MelodicObject.MelodicObject method), 1088
- `__le__()` (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet.NamedChromaticPitch.NamedChromaticPitch method), 1091
- `__le__()` (abjad.tools.pitchtools.IntervalObject.IntervalObject.IntervalObject.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1093
- `__le__()` (abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass.IntervalObjectClass.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1094
- `__le__()` (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment.IntervalObjectSegment.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1096
- `__le__()` (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment method), 1098
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.NamedChromaticIntervalClass.NamedChromaticIntervalClass method), 1158
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment.NamedChromaticIntervalClassSegment.NamedChromaticIntervalClassSegment method), 1160
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.NamedChromaticIntervalClassSet.NamedChromaticIntervalClassSet method), 1163
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.NamedChromaticIntervalClassVector.NamedChromaticIntervalClassVector method), 1166
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.NamedDiatonicIntervalClass.NamedDiatonicIntervalClass method), 1168
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment.NamedDiatonicIntervalClassSegment.NamedDiatonicIntervalClassSegment method), 1170
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.NamedDiatonicIntervalClassVector.NamedDiatonicIntervalClassVector method), 1173
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticIntervalPitchClassSegment.NumericalPitchClassSegment method), 1175
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClassSegment.NumericalPitchClassSegment method), 1177
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.NumericalPitchClassSegment method), 1179
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.NumericalPitchClassSegment method), 1182
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.NumericalPitchClassSegment method), 1185
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.NumericalPitchClassSegment method), 1188
- `__le__()` (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval.NumericalPitchClassSegment method), 1190
- `__le__()` (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.NumericalPitchClassSegment method), 1192
- `__le__()` (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicIntervalPitchClassSegment.NumericalPitchClassSegment method), 1194
- `__le__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClassSegment.NumericalPitchClassSegment method), 1196
- `__le__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.NumericalPitchClassSegment method), 1198
- `__le__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.NumericalPitchClassSegment method), 1201
- `__le__()` (abjad.tools.pitchtools.MelodicIntervalClassObject.MelodicIntervalClassObject.MelodicIntervalClassObject.MelodicIntervalClassObject.NumericalPitchClassSegment method), 1101

__le__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1258

__le__() (abjad.tools.pitchtools.PitchClassObject.PitchClassObject.PitchClassObject method), 1117

__le__() (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment.PitchClassObjectSegment method), 1119

__le__() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1121

__le__() (abjad.tools.pitchtools.PitchObject.PitchObject.PitchObject method), 1124

__le__() (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment.PitchObjectSegment method), 1125

__le__() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet method), 1128

__le__() (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange method), 1262

__le__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1264

__le__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 1268

__le__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest method), 1310

__le__() (abjad.tools.resttools.Rest.Rest.Rest method), 1313

__le__() (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer method), 1336

__le__() (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf method), 1342

__le__() (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode method), 1322

__le__() (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser method), 1346

__le__() (abjad.tools.schemetools.Scheme.Scheme.Scheme method), 1348

__le__() (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList method), 1350

__le__() (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor method), 1351

__le__() (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment method), 1353

__le__() (abjad.tools.schemetools.SchemePair.SchemePair.SchemePair method), 1354

__le__() (abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector method), 1356

__le__() (abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant.SchemeVectorConstant method), 1358

__le__() (abjad.tools.scoretemplatetools.GroupedRhythmicStatesScoreTemplate.GroupedRhythmicStatesScoreTemplate method), 1361

__le__() (abjad.tools.scoretemplatetools.GroupedStavesScoreTemplate.GroupedStavesScoreTemplate method), 1363

__le__() (abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplate.ScoreTemplate method), 1360

__le__() (abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.StringQuartetScoreTemplate method), 1365

__le__() (abjad.tools.scoretools.CyclicList.CyclicList.CyclicList method), 1419

__le__() (abjad.tools.scoretools.CyclicMatrix.CyclicMatrix.CyclicMatrix method), 1422

__le__() (abjad.tools.scoretools.CyclicTree.CyclicTree.CyclicTree method), 1431

__le__() (abjad.tools.scoretools.CyclicTuple.CyclicTuple.CyclicTuple method), 1433

__le__() (abjad.tools.scoretools.Matrix.Matrix.Matrix method), 1435

__le__() (abjad.tools.scoretools.Tree.Tree.Tree method), 1443

__le__() (abjad.tools.scoretools.ResidueClass.ResidueClass.ResidueClass method), 1491

__le__() (abjad.tools.scoretools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression method), 1493

__le__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1497

__le__() (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner method), 1518

__le__() (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner method), 1525

__le__() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1533

__le__() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1542

__le__() (abjad.tools.spannertools.DirectedSpanner.DirectedSpanner.DirectedSpanner method), 1505

__le__() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1549

__le__() (abjad.tools.spannertools.GroupedRhythmicStatesSpanner.GroupedRhythmicStatesSpanner.GroupedRhythmicStatesSpanner method), 1555

__le__() (abjad.tools.spannertools.GroupedStavesSpanner.GroupedStavesSpanner.GroupedStavesSpanner method), 1563

__le__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1569

__le__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1576

- `__le__()` (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck.DuplicateIdCheck.ImmutableDictionary.ImmutableDictionary method), 2086
- `__le__()` (abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck.EmptyContainerCheck.ObjectInventory.ObjectInventory method), 2087
- `__le__()` (abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck.IntermarkedHairpinCheck.InheritanceGrammar.InheritanceGrammar method), 2089
- `__le__()` (abjad.tools.wellformednesstools.MisduratedMeasureCheck.MisduratedMeasureCheck.MisduratedMeasureCheck.Inner.GraceContainer.GraceContainer method), 2090
- `__le__()` (abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck.MisfilledMeasureCheck.InstrumentInventory.InstrumentInventory method), 2092
- `__le__()` (abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck.MispitchedTieCheck.BookBlock.BookBlock.BookBlock method), 2093
- `__le__()` (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck.PartBlock.PartBlock method), 2095
- `__le__()` (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.MissingParentCheck.LilyPondFile.LilyPondFile.LilyPondFile method), 2096
- `__le__()` (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck.NonattributedBlock.NonattributedBlock method), 2098
- `__le__()` (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck.ScoreBlock.ScoreBlock method), 2099
- `__le__()` (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck.SyntaxNode.SyntaxNode method), 2101
- `__le__()` (abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.OverlappingOctavationCheck.Music.Music method), 2102
- `__le__()` (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck.Ratio.Ratio.Ratio method), 2104
- `__len__()` (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection.tools.measuretools.AnonymousMeasure.AnonymousMeasure method), 1881
- `__len__()` (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 229
- `__len__()` (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner.Measure.Measure.Measure method), 237
- `__len__()` (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.Pitch.Pitch method), 246
- `__len__()` (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.HarmonicIntervalSegment.HarmonicIntervalSegment method), 254
- `__len__()` (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.ChromaticIntervalSegment.ChromaticIntervalSegment method), 261
- `__len__()` (abjad.tools.chordtools.Chord.Chord.Chord __len__ (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 271
- `__len__()` (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig.abjadtools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1897
- `__len__()` (abjad.tools.containertools.Cluster.Cluster.Cluster __len__ (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 341
- `__len__()` (abjad.tools.containertools.Container.Container.Container __len__ (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 347
- `__len__()` (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer.IntervalClassObjectSegment.IntervalClassObjectSegment method), 355
- `__len__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.IntervalClassObjectSet.IntervalClassObjectSet method), 382
- `__len__()` (abjad.tools.contexttools.Context.Context.Context __len__ (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment method), 391
- `__len__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet method), 416

__len__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method), 1610
 __len__() (abjad.tools.spannertools.Spanner.Spanner.Spanner method), 1511
 __len__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner method), 1617
 __len__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner method), 1623
 __len__() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner method), 1630
 __len__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method), 1637
 __len__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1661
 __len__() (abjad.tools.stafftools.Staff.Staff.Staff method), 1670
 __len__() (abjad.tools.tietools.TieChain.TieChain.TieChain method), 1683
 __len__() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner method), 1690
 __len__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1709
 __len__() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method), 1722
 __len__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1744
 __len__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1805
 __len__() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 1809
 __len__() (abjad.tools.tonalitytools.Mode.Mode.Mode method), 1812
 __len__() (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1818
 __len__() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1838
 __len__() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1847
 __len__() (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment method), 1861
 __len__() (abjad.tools.voicetools.Voice.Voice.Voice method), 1872
 __lt__() (abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject method), 1875
 __lt__() (abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject method), 1877
 __lt__() (abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject method), 1878
 __lt__() (abjad.tools.abctools.Parser.Parser.Parser method), 1880
 __lt__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection method), 1881
 __lt__() (abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject method), 1883
 __len__() (abjad.tools.abjadbooktools.AbjadBookProcessor.AbjadBookProcessor.AbjadBookProcessor method), 1886
 __len__() (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.AbjadBookScript method), 1888
 __len__() (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock method), 1889
 __len__() (abjad.tools.abjadbooktools.HTMLOutputFormat.HTMLOutputFormat.HTMLOutputFormat method), 1891
 __len__() (abjad.tools.abjadbooktools.LaTeXOutputFormat.LaTeXOutputFormat.LaTeXOutputFormat method), 1892
 __len__() (abjad.tools.abjadbooktools.OutputFormat.OutputFormat.OutputFormat method), 1884
 __len__() (abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutputFormat.ReSTOutputFormat method), 1894
 __len__() (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner method), 229
 __len__() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 237
 __len__() (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 246
 __len__() (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 254
 __len__() (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 261
 __len__() (abjad.tools.beamtools.ChordTools.ChordTools.ChordTools method), 271
 __len__() (abjad.tools.componenttools.Component.Component.Component method), 277
 __len__() (abjad.tools.componenttools.ContainmentSignature.ContainmentSignature.ContainmentSignature method), 279
 __len__() (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1897
 __len__() (abjad.tools.containertools.Cluster.Cluster.Cluster method), 341
 __len__() (abjad.tools.containertools.Container.Container.Container method), 348
 __len__() (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer method), 355
 __len__() (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark method), 380
 __len__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 383
 __len__() (abjad.tools.contexttools.Context.Context.Context method), 391
 __len__() (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark method), 394
 __len__() (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark method), 397
 __len__() (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark method), 401
 __len__() (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark method), 405
 __len__() (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark method), 409

- __lt__() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method), 414
- __lt__() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 416
- __lt__() (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark method), 421
- __lt__() (abjad.tools.datastructuretools.Digraph.Digraph.Digraph method), 1902
- __lt__() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method), 1904
- __lt__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1907
- __lt__() (abjad.tools.datastructuretools.OrdinalConstant.OrdinalConstant.OrdinalConstant method), 1909
- __lt__() (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript method), 1916
- __lt__() (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript method), 1919
- __lt__() (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript method), 1921
- __lt__() (abjad.tools.developerscripttools.CleanScript.CleanScript.CleanScript method), 1924
- __lt__() (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScript method), 1926
- __lt__() (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScript method), 1928
- __lt__() (abjad.tools.developerscripttools.DeveloperScript.DeveloperScript.DeveloperScript method), 1911
- __lt__() (abjad.tools.developerscripttools.DirectoryScript.DirectoryScript.DirectoryScript method), 1914
- __lt__() (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript method), 1931
- __lt__() (abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript method), 1933
- __lt__() (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript.RenameModulesScript method), 1935
- __lt__() (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript method), 1938
- __lt__() (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript.ReplacePromptsScript method), 1941
- __lt__() (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScript method), 1943
- __lt__() (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript method), 1945
- __lt__() (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript method), 1948
- __lt__() (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript.SvnMessageScript method), 1950
- __lt__() (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript method), 1953
- __lt__() (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler method), 1954
- __lt__() (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator method), 1956
- __lt__() (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler method), 1958
- __lt__() (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter method), 1959
- __lt__() (abjad.tools.documentationtools.Documenter.Documenter.Documenter method), 1961
- __lt__() (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler method), 1963
- __lt__() (abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter method), 1964
- __lt__() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1968
- __lt__() (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler method), 1969
- __lt__() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1971
- __lt__() (abjad.tools.durationtools.Duration.Duration.Duration method), 447
- __lt__() (abjad.tools.durationtools.Offset.Offset.Offset method), 451
- __lt__() (abjad.tools.durationtools.TimespanConstant.TimespanConstant.TimespanConstant method), 454
- __lt__() (abjad.tools.durationtools.Container.GraceContainer.GraceContainer.GraceContainer method), 482
- __lt__() (abjad.tools.durationtools.Accordian.Accordian.Accordian method), 491
- __lt__() (abjad.tools.durationtools.AltoFlute.AltoFlute.AltoFlute method), 497
- __lt__() (abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 503
- __lt__() (abjad.tools.durationtools.AltoTrumpet.AltoTrumpet.AltoTrumpet method), 509
- __lt__() (abjad.tools.durationtools.BassClarinet.BassClarinet.BassClarinet method), 515
- __lt__() (abjad.tools.durationtools.BassFlute.BassFlute.BassFlute method), 521
- __lt__() (abjad.tools.durationtools.BassSaxophone.BassSaxophone.BassSaxophone method), 527
- __lt__() (abjad.tools.durationtools.BassTrombone.BassTrombone.BassTrombone method), 533
- __lt__() (abjad.tools.durationtools.BassVoice.BassVoice.BassVoice method), 539
- __lt__() (abjad.tools.durationtools.Bassoon.Bassoon.Bassoon method), 545
- __lt__() (abjad.tools.durationtools.Cello.Cello.Cello method), 551
- __lt__() (abjad.tools.durationtools.ClarinetInA.ClarinetInA.ClarinetInA method), 557
- __lt__() (abjad.tools.durationtools.ClarinetInBb.ClarinetInBb.ClarinetInBb method), 563
- __lt__() (abjad.tools.instrumenttools.Cello.Cello.Cello method), 569
- __lt__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 575

- `__lt__()` (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 581 method), 739
- `__lt__()` (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 587 method), 745
- `__lt__()` (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 593 method), 751
- `__lt__()` (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone (abjad.tools.instrumenttools.Viola.Viola.Viola method), 599 method), 757
- `__lt__()` (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon (abjad.tools.instrumenttools.Violin.Violin.Violin method), 605 method), 763
- `__lt__()` (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 611 method), 769
- `__lt__()` (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 617 method), 824
- `__lt__()` (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 623 method), 830
- `__lt__()` (abjad.tools.instrumenttools.Flute.Flute.Flute (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken method), 629 method), 862
- `__lt__()` (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 634 method), 865
- `__lt__()` (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 640 method), 868
- `__lt__()` (abjad.tools.instrumenttools.Guitar.Guitar.Guitar (abjad.tools.lilypondfiletools.DateTimeToken.DateTimeToken.DateTimeToken method), 646 method), 871
- `__lt__()` (abjad.tools.instrumenttools.Harp.Harp.Harp (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 652 method), 877
- `__lt__()` (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 658 method), 879
- `__lt__()` (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken method), 660 method), 880
- `__lt__()` (abjad.tools.instrumenttools.Marimba.Marimba.Marimba (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 667 method), 860
- `__lt__()` (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice (abjad.tools.lilypondparsertools.ScoreBlock.ScoreBlock.ScoreBlock method), 673 method), 885
- `__lt__()` (abjad.tools.instrumenttools.Oboe.Oboe.Oboe (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy method), 679 method), 2036
- `__lt__()` (abjad.tools.instrumenttools.Piano.Piano.Piano (abjad.tools.lilypondparsertools.LilyPondDuration.LilyPondDuration.LilyPondDuration method), 685 method), 2037
- `__lt__()` (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo (abjad.tools.lilypondparsertools.LilyPondEvent.LilyPondEvent.LilyPondEvent method), 691 method), 2038
- `__lt__()` (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone (abjad.tools.lilypondparsertools.LilyPondFraction.LilyPondFraction.LilyPondFraction method), 697 method), 2040
- `__lt__()` (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition.LilyPondLexicalDefinition method), 703 method), 2043
- `__lt__()` (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser method), 709 method), 2048
- `__lt__()` (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 715 method), 2067
- `__lt__()` (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser method), 721 method), 2073
- `__lt__()` (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser method), 727 method), 2077
- `__lt__()` (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet (abjad.tools.lilypondparsertools.SyntaxNode.SyntaxNode.SyntaxNode method), 733 method), 2079

__lt__() (abjad.tools.marktools.Annotation.Annotation.Annotation method), 893

__lt__() (abjad.tools.marktools.Articulation.Articulation.Articulation method), 895

__lt__() (abjad.tools.marktools.BarLine.BarLine.BarLine method), 899

__lt__() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 902

__lt__() (abjad.tools.marktoolsDirectedMark.DirectedMark.DirectedMark method), 890

__lt__() (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 905

__lt__() (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method), 908

__lt__() (abjad.tools.marktools.Mark.Mark.Mark method), 910

__lt__() (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method), 913

__lt__() (abjad.tools.markuptools.Markup.Markup.Markup method), 934

__lt__() (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand method), 936

__lt__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 939

__lt__() (abjad.tools.mathtools.BoundedObject.BoundedObject.BoundedObject method), 946

__lt__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction method), 951

__lt__() (abjad.tools.mathtools.Ratio.Ratio.Ratio method), 955

__lt__() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 987

__lt__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 999

__lt__() (abjad.tools.measuretools.Measure.Measure.Measure method), 1009

__lt__() (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 1036

__lt__() (abjad.tools.notetools.Note.Note.Note method), 1040

__lt__() (abjad.tools.notetools.NoteHead.NoteHead.NoteHead method), 1043

__lt__() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method), 1051

__lt__() (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell method), 1054

__lt__() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1056

__lt__() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1058

__lt__() (abjad.tools.pitchtools.Accidental.Accidental.Accidental method), 1130

__lt__() (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject.ChromaticIntervalClassObject method), 1065

__lt__() (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.ChromaticIntervalObject method), 1067

__lt__() (abjad.tools.pitchtools.ChromaticObject.ChromaticObject.ChromaticObject method), 1068

__lt__() (abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject.ChromaticPitchObject method), 1070

__lt__() (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject.CounterpointIntervalClassObject method), 1071

__lt__() (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.CounterpointIntervalObject method), 1073

__lt__() (abjad.tools.pitchtools.CounterpointMark.ChromaticIntervalClassObject.CounterpointMark.ChromaticIntervalClassObject method), 1074

__lt__() (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject.DiatonicIntervalClassObject method), 1076

__lt__() (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.DiatonicIntervalObject method), 1077

__lt__() (abjad.tools.pitchtools.DiatonicObject.DiatonicObject.DiatonicObject method), 1078

__lt__() (abjad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClassObject.DiatonicPitchClassObject method), 1080

__lt__() (abjad.tools.pitchtools.DiatonicPitchObject.DiatonicPitchObject.DiatonicPitchObject method), 1082

__lt__() (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval method), 1132

__lt__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 1133

__lt__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1136

__lt__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1138

__lt__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1141

__lt__() (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 1143

__lt__() (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1145

__lt__() (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval method), 1146

__lt__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1148

__lt__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1151

__lt__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1153

__lt__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1156

__lt__() (abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject.HarmonicIntervalClassObject method), 1083

__lt__() (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.HarmonicIntervalObject method), 1085

__lt__() (abjad.tools.pitchtools.HarmonicObject.HarmonicObject.HarmonicObject method), 1086

__lt__() (abjad.tools.pitchtools.HarmonicIntervalClassObject.ChromaticIntervalClassObject.ChromaticIntervalClassObject method), 1088

[illegible]

- `__lt__()` (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment.PitchClassObjectSegmentSpecifier.InstrumentationSp method), 1119
- `__lt__()` (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet.Performer.Performer.Performer method), 1122
- `__lt__()` (abjad.tools.pitchtools.PitchObject.PitchObject.PitchObject) (abjad.tools.scoretools.PerformerInventory.PerformerInventory.Per method), 1124
- `__lt__()` (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment.PitchObjectSegment.PianoStaff.PianoStaff.PianoStaff method), 1125
- `__lt__()` (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet) (abjad.tools.scoretools.Score.Score.Score method), 1128
- `__lt__()` (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange) (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1262
- `__lt__()` (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory) (abjad.tools.scoretools.CyclicList.CyclicList.CyclicList method), 1264
- `__lt__()` (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow) (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix.CyclicMatrix method), 1268
- `__lt__()` (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest) (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1310
- `__lt__()` (abjad.tools.resttools.Rest.Rest.Rest) (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple method), 1313
- `__lt__()` (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer) (abjad.tools.sequencetools.Matrix.Matrix.Matrix method), 1336
- `__lt__()` (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf) (abjad.tools.sequencetools.Tree.Tree.Tree method), 1342
- `__lt__()` (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode) (abjad.tools.sequencetools.ResidueClass.ResidueClass.ResidueClass method), 1322
- `__lt__()` (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser) (abjad.tools.sequencetools.ResidueClassExpression.ResidueClassExpression method), 1346
- `__lt__()` (abjad.tools.schemetools.Scheme.Scheme.Scheme) (abjad.tools.skiptools.Skip.Skip.Skip method), 1348
- `__lt__()` (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList) (abjad.tools.skiptools.BracketSpanner.BracketSpanner.BracketSpanner method), 1350
- `__lt__()` (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor) (abjad.tools.skiptools.ComplexGlissandoSpanner.ComplexGlissandoSpanner method), 1351
- `__lt__()` (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment) (abjad.tools.skiptools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1353
- `__lt__()` (abjad.tools.schemetools.SchemePair.SchemePair.SchemePair) (abjad.tools.skiptools.DecrescendoSpanner.DecrescendoSpanner method), 1354
- `__lt__()` (abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector) (abjad.tools.skiptools.DirectedSpanner.DirectedSpanner.DirectedSpanner method), 1356
- `__lt__()` (abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant.SchemeVectorConstant) (abjad.tools.skiptools.DynamicTextSpanner.DynamicTextSpanner method), 1358
- `__lt__()` (abjad.tools.scoretemplatetools.GroupedRhythmicStavesScoreTemplate.GroupedRhythmicStavesScoreTemplate) (abjad.tools.skiptools.GroupedRhythmicStavesScoreTemplate method), 1362
- `__lt__()` (abjad.tools.scoretemplatetools.GroupedStavesScoreTemplate.GroupedStavesScoreTemplate) (abjad.tools.skiptools.GroupedStavesScoreTemplate method), 1363
- `__lt__()` (abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplate.ScoreTemplate) (abjad.tools.skiptools.HiddenStaffSpanner.HiddenStaffSpanner method), 1360
- `__lt__()` (abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.StringQuartetScoreTemplate) (abjad.tools.skiptools.StringQuartetScoreTemplate method), 1366
- `__lt__()` (abjad.tools.scoretemplatetools.TwoStaffPianoScoreTemplate.TwoStaffPianoScoreTemplate) (abjad.tools.skiptools.TwoStaffPianoScoreTemplate method), 1367
- `__lt__()` (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff) (abjad.tools.skiptools.OctavationSpanner.OctavationSpanner.OctavationSpanner method), 1376

__lt__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker method), 1597

__lt__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner.TimeTokenMaker.TimeTokenMaker.TimeTokenMaker method), 1604

__lt__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner abjad.tools.timetoken tools.TokenBurnishedSignalFilledTimeTokenMaker method), 1610

__lt__() (abjad.tools.spannertools.Spanner.Spanner.Spanner __lt__() (abjad.tools.timetoken tools.TokenIncisedNoteFilledTimeTokenMaker method), 1511

__lt__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner abjad.tools.timetoken tools.TokenIncisedRestFilledTimeTokenMaker method), 1617

__lt__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner abjad.tools.timetoken tools.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker method), 1624

__lt__() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner abjad.tools.timetoken tools.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker method), 1630

__lt__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1637

__lt__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 1661

__lt__() (abjad.tools.stafftools.Staff.Staff.Staff method), __lt__() (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator method), 1670

__lt__() (abjad.tools.tietools.TieChain.TieChain.TieChain __lt__() (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator method), 1683

__lt__() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner __lt__() (abjad.tools.tonalitytools.Mode.Mode.Mode method), 1690

__lt__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator method), 1709

__lt__() (abjad.tools.timeintervaltools.TimeIntervalAggregate.Mixin abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator method), 1703

__lt__() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator method), 1706

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator method), 1722

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator method), 1744

__lt__() (abjad.tools.timetoken tools.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker.TonalFunction.TonalFunction.TonalFunction method), 1759

__lt__() (abjad.tools.timetoken tools.EqualDivisionTimeTokenMaker.EqualDivisionTimeTokenMaker.EqualDivisionTimeTokenMaker.EqualDivisionTimeTokenMaker method), 1770

__lt__() (abjad.tools.timetoken tools.IncisedTimeTokenMaker.IncisedTimeTokenMaker.IncisedTimeTokenMaker.IncisedTimeTokenMaker method), 1761

__lt__() (abjad.tools.timetoken tools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker method), 1773

__lt__() (abjad.tools.timetoken tools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker method), 1776

__lt__() (abjad.tools.timetoken tools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker method), 1779

__lt__() (abjad.tools.timetoken tools.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker method), 1782

__lt__() (abjad.tools.timetoken tools.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker method), 1763

__lt__() (abjad.tools.timetoken tools.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker method), 1785

__lt__() (abjad.tools.timetoken tools.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker method), 1788

__lt__() (abjad.tools.wellformednesstools.IntermarkedHairpinCheck, (abjad.tools.wellformednesstools.IntermarkedHairpinCheck, method), 2089

__lt__() (abjad.tools.wellformednesstools.MisdurationMeasureCheck, (abjad.tools.wellformednesstools.MisdurationMeasureCheck, method), 2090

__lt__() (abjad.tools.wellformednesstools.MisfilledMeasureCheck, (abjad.tools.wellformednesstools.MisfilledMeasureCheck, method), 2092

__lt__() (abjad.tools.wellformednesstools.MispitchedTieCheck, (abjad.tools.wellformednesstools.MispitchedTieCheck, method), 2093

__lt__() (abjad.tools.wellformednesstools.MisrepresentedFlagCheck, (abjad.tools.wellformednesstools.MisrepresentedFlagCheck, method), 2095

__lt__() (abjad.tools.wellformednesstools.MissingParentCheck, (abjad.tools.wellformednesstools.MissingParentCheck, method), 2096

__lt__() (abjad.tools.wellformednesstools.NestedMeasureCheck, (abjad.tools.wellformednesstools.NestedMeasureCheck, method), 2098

__lt__() (abjad.tools.wellformednesstools.OverlappingBeamCheck, (abjad.tools.wellformednesstools.OverlappingBeamCheck, method), 2099

__lt__() (abjad.tools.wellformednesstools.OverlappingGlissandoCheck, (abjad.tools.wellformednesstools.OverlappingGlissandoCheck, method), 2101

__lt__() (abjad.tools.wellformednesstools.OverlappingOctavationCheck, (abjad.tools.wellformednesstools.OverlappingOctavationCheck, method), 2102

__lt__() (abjad.tools.wellformednesstools.ShortHairpinCheck, (abjad.tools.wellformednesstools.ShortHairpinCheck, method), 2104

__mod__() (abjad.tools.durationtools.Duration.Duration.Duration, (abjad.tools.durationtools.Duration.Duration.Duration, method), 447

__mod__() (abjad.tools.durationtools.Offset.Offset.Offset, (abjad.tools.durationtools.Offset.Offset.Offset, method), 451

__mod__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction, (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction, method), 951

__mul__() (abjad.tools.chordtools.Chord.Chord.Chord, (abjad.tools.chordtools.Chord.Chord.Chord, method), 271

__mul__() (abjad.tools.componenttools.Component.Component, (abjad.tools.componenttools.Component.Component, method), 277

__mul__() (abjad.tools.containertools.Cluster.Cluster.Cluster, (abjad.tools.containertools.Cluster.Cluster.Cluster, method), 341

__mul__() (abjad.tools.containertools.Container.Container.Container, (abjad.tools.containertools.Container.Container.Container, method), 348

__mul__() (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer, (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer, method), 355

__mul__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory, (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory, method), 383

__mul__() (abjad.tools.contexttools.Context.Context.Context, (abjad.tools.contexttools.Context.Context.Context, method), 391

__mul__() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark, (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark, method), 414

__mul__() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory, (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory, method), 416

__mul__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory, (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory, method), 1907

__mul__() (abjad.tools.durationtools.Duration.Duration.Duration, (abjad.tools.durationtools.Duration.Duration.Duration, method), 447

__mul__() (abjad.tools.durationtools.Offset.Offset.Offset, (abjad.tools.durationtools.Offset.Offset.Offset, method), 451

__mul__() (abjad.tools.gracetools.GraceContainer.GraceContainer, (abjad.tools.gracetools.GraceContainer.GraceContainer, method), 482

__mul__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure, (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure, method), 999

__mul__() (abjad.tools.measuretools.Measure.Measure.Measure, (abjad.tools.measuretools.Measure.Measure.Measure, method), 1009

__mul__() (abjad.tools.note tools.Note.Note.Note, (abjad.tools.note tools.Note.Note.Note, method), 1040

__mul__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment, (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment, method), 1139

__mul__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment, (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment, method), 1153

__mul__() (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment, (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment, method), 1088

__mul__() (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment, (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment, method), 1096

__mul__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment, (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment, method), 1160

__mul__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment, (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment, method), 1171

__mul__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment, (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment, method), 1179

__mul__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment, (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment, method), 1185

__mul__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval, (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval, method), 1194

__mul__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment, (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment, method), 1198

__mul__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment, (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment, method), 1211

__mul__() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment, (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment, method), 1217

<code>__mul__()</code>	<code>(abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.method), 1238</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.method), 1238</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.method), 1238</code>
<code>__mul__()</code>	<code>(abjad.tools.pitchtools.ObjectSegment.ObjectSegment.method), 1110</code>	<code>__mul__()</code>	<code>(abjad.tools.abctools.Parser.Parser.Parser.method), 1880</code>	<code>__mul__()</code>	<code>(abjad.tools.abctools.Parser.Parser.Parser.method), 1880</code>
<code>__mul__()</code>	<code>(abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.method), 1253</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.method), 1253</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.method), 1253</code>
<code>__mul__()</code>	<code>(abjad.tools.pitchtools.OctaveTranspositionMapping.InvertedOctaveTranspositionMapping.method), 1259</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.OctaveTranspositionMapping.InvertedOctaveTranspositionMapping.method), 1259</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.OctaveTranspositionMapping.InvertedOctaveTranspositionMapping.method), 1259</code>
<code>__mul__()</code>	<code>(abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment.method), 1119</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment.method), 1119</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment.method), 1119</code>
<code>__mul__()</code>	<code>(abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment.method), 1126</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment.method), 1126</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment.method), 1126</code>
<code>__mul__()</code>	<code>(abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.method), 1264</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.method), 1264</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.method), 1264</code>
<code>__mul__()</code>	<code>(abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.method), 1268</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.method), 1268</code>	<code>__mul__()</code>	<code>(abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.method), 1268</code>
<code>__mul__()</code>	<code>(abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.method), 1310</code>	<code>__mul__()</code>	<code>(abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.method), 1310</code>	<code>__mul__()</code>	<code>(abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.method), 1310</code>
<code>__mul__()</code>	<code>(abjad.tools.resttools.Rest.Rest.Rest.method), 1313</code>	<code>__ne__()</code>	<code>(abjad.tools.abjadbooktools.OutputFormat.OutputFormat.OutputFormat.method), 1884</code>	<code>__mul__()</code>	<code>(abjad.tools.abjadbooktools.OutputFormat.OutputFormat.OutputFormat.method), 1884</code>
<code>__mul__()</code>	<code>(abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff.method), 1376</code>	<code>__mul__()</code>	<code>(abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutputFormat.ReSTOutputFormat.method), 1894</code>	<code>__mul__()</code>	<code>(abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutputFormat.ReSTOutputFormat.method), 1894</code>
<code>__mul__()</code>	<code>(abjad.tools.scoretools.PerformerInventory.PerformerInventory.method), 1384</code>	<code>__mul__()</code>	<code>(abjad.tools.beamttools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner.method), 237</code>	<code>__mul__()</code>	<code>(abjad.tools.beamttools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner.method), 237</code>
<code>__mul__()</code>	<code>(abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff.method), 1393</code>	<code>__mul__()</code>	<code>(abjad.tools.beamttools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.method), 246</code>	<code>__mul__()</code>	<code>(abjad.tools.beamttools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.method), 246</code>
<code>__mul__()</code>	<code>(abjad.tools.scoretools.Score.Score.Score.method), 1402</code>	<code>__mul__()</code>	<code>(abjad.tools.beamttools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.method), 254</code>	<code>__mul__()</code>	<code>(abjad.tools.beamttools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.method), 254</code>
<code>__mul__()</code>	<code>(abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup.method), 1411</code>	<code>__mul__()</code>	<code>(abjad.tools.beamttools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.method), 261</code>	<code>__mul__()</code>	<code>(abjad.tools.beamttools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.method), 261</code>
<code>__mul__()</code>	<code>(abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList.method), 1419</code>	<code>__mul__()</code>	<code>(abjad.tools.chordtools.Chord.Chord.Chord.method), 271</code>	<code>__mul__()</code>	<code>(abjad.tools.chordtools.Chord.Chord.Chord.method), 271</code>
<code>__mul__()</code>	<code>(abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple.method), 1433</code>	<code>__mul__()</code>	<code>(abjad.tools.componenttools.Component.Component.Component.method), 277</code>	<code>__mul__()</code>	<code>(abjad.tools.componenttools.Component.Component.Component.method), 277</code>
<code>__mul__()</code>	<code>(abjad.tools.skiptools.Skip.Skip.Skip.method), 1497</code>	<code>__mul__()</code>	<code>(abjad.tools.componenttools.ContainmentSignature.ContainmentSignature.ContainmentSignature.method), 279</code>	<code>__mul__()</code>	<code>(abjad.tools.componenttools.ContainmentSignature.ContainmentSignature.ContainmentSignature.method), 279</code>
<code>__mul__()</code>	<code>(abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff.method), 1662</code>	<code>__mul__()</code>	<code>(abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig.method), 1897</code>	<code>__mul__()</code>	<code>(abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig.method), 1897</code>
<code>__mul__()</code>	<code>(abjad.tools.stafftools.Staff.Staff.Staff.method), 1671</code>	<code>__mul__()</code>	<code>(abjad.tools.clustertools.Cluster.Cluster.Cluster.method), 341</code>	<code>__mul__()</code>	<code>(abjad.tools.clustertools.Cluster.Cluster.Cluster.method), 341</code>
<code>__mul__()</code>	<code>(abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator.method), 1809</code>	<code>__mul__()</code>	<code>(abjad.tools.containertools.Container.Container.Container.method), 348</code>	<code>__mul__()</code>	<code>(abjad.tools.containertools.Container.Container.Container.method), 348</code>
<code>__mul__()</code>	<code>(abjad.tools.tonalitytools.Scale.Scale.Scale.method), 1818</code>	<code>__mul__()</code>	<code>(abjad.tools.contexttools.ClefMark.ClefMark.ClefMark.method), 380</code>	<code>__mul__()</code>	<code>(abjad.tools.contexttools.ClefMark.ClefMark.ClefMark.method), 380</code>
<code>__mul__()</code>	<code>(abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.method), 1838</code>	<code>__mul__()</code>	<code>(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.method), 383</code>	<code>__mul__()</code>	<code>(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.method), 383</code>
<code>__mul__()</code>	<code>(abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet.method), 1847</code>	<code>__ne__()</code>	<code>(abjad.tools.abjadbooktools.Context.Context.Context.method), 391</code>	<code>__ne__()</code>	<code>(abjad.tools.abjadbooktools.Context.Context.Context.method), 391</code>
<code>__ne__()</code>	<code>(abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject.method), 1875</code>	<code>__ne__()</code>	<code>(abjad.tools.abjadbooktools.ContextMark.ContextMark.ContextMark.method), 394</code>	<code>__ne__()</code>	<code>(abjad.tools.abjadbooktools.ContextMark.ContextMark.ContextMark.method), 394</code>
<code>__ne__()</code>	<code>(abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.method), 1877</code>	<code>__ne__()</code>	<code>(abjad.tools.abjadbooktools.ContextMark.ContextMark.ContextMark.method), 394</code>	<code>__ne__()</code>	<code>(abjad.tools.abjadbooktools.ContextMark.ContextMark.ContextMark.method), 394</code>

- `__ne__()` (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMarkTools.developerscripttools.SvnMessageScript.SvnMessageScript method), 397
- `__ne__()` (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMarkTools.developerscripttools.SvnUpdateScript.SvnUpdateScript method), 401
- `__ne__()` (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMarkTools.APICrawler.APICrawler.APICrawler method), 405
- `__ne__()` (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMarkTools.AbjadAPIGenerator.AbjadAPIGenerator method), 409
- `__ne__()` (abjad.tools.contexttools.TempoMark.TempoMark.TempoMarkTools.abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler method), 414
- `__ne__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventoryTools.abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter method), 417
- `__ne__()` (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMarkTools.Documenter.Documenter.Documenter method), 421
- `__ne__()` (abjad.tools.datastructuretools.Digraph.Digraph.Digraph) (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler method), 1902
- `__ne__()` (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionaryTools.FunctionDocumenter.FunctionDocumenter method), 1904
- `__ne__()` (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventoryTools.abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph method), 1907
- `__ne__()` (abjad.tools.datastructuretools.OrdinalConstant.OrdinalConstant.OrdinalConstantTools.abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler method), 1909
- `__ne__()` (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScriptTools.abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1917
- `__ne__()` (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScriptTools.abjad.tools.documentationtools.Duration.Duration.Duration method), 1919
- `__ne__()` (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScriptTools.abjad.tools.documentationtools.Offset.Offset.Offset method), 1921
- `__ne__()` (abjad.tools.developerscripttools.CleanScript.CleanScript.CleanScriptTools.abjad.tools.durationtools.TimespanConstant.TimespanConstant.TimespanConstant method), 1924
- `__ne__()` (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScriptTools.abjad.tools.durationtools.Container.GraceContainer.GraceContainer method), 1926
- `__ne__()` (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScriptTools.abjad.tools.durationtools.Accordion.Accordion.Accordion method), 1928
- `__ne__()` (abjad.tools.developerscripttools.DeveloperScript.DeveloperScript.DeveloperScriptTools.abjad.tools.durationtools.AltoFlute.AltoFlute.AltoFlute method), 1912
- `__ne__()` (abjad.tools.developerscripttools.DirectoryScript.DirectoryScript.DirectoryScriptTools.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 1914
- `__ne__()` (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScriptTools.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 1931
- `__ne__()` (abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScriptTools.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 1933
- `__ne__()` (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript.RenameModulesScriptTools.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 1935
- `__ne__()` (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScriptTools.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 1938
- `__ne__()` (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript.ReplacePromptsScriptTools.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 1941
- `__ne__()` (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScriptTools.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 1943
- `__ne__()` (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScriptTools.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 1945
- `__ne__()` (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScriptTools.abjad.tools.durationtools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 1948

__ne__() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 557

__ne__() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 563

__ne__() (abjad.tools.instrumenttools.Cello.Cello.Cello method), 569

__ne__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 575

__ne__() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 581

__ne__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 587

__ne__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 593

__ne__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 599

__ne__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 605

__ne__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 611

__ne__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 617

__ne__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 623

__ne__() (abjad.tools.instrumenttools.Flute.Flute.Flute method), 629

__ne__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 634

__ne__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 640

__ne__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 646

__ne__() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 652

__ne__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 658

__ne__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 661

__ne__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 667

__ne__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 673

__ne__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 679

__ne__() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 685

__ne__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 691

__ne__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 697

__ne__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 703

__ne__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 709

__ne__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 715

__ne__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 721

__ne__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 727

__ne__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 733

__ne__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 739

__ne__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 745

__ne__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 751

__ne__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 757

__ne__() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 763

__ne__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 769

__ne__() (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 825

__ne__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 830

__ne__() (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 862

__ne__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 865

__ne__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 868

__ne__() (abjad.tools.lilypondfiletools.DateTimeToken.DateTimeToken.DateTimeToken method), 871

__ne__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 877

__ne__() (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 879

__ne__() (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken method), 880

__ne__() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 861

__ne__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 885

__ne__() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy method), 2036

__ne__() (abjad.tools.lilypondparsertools.LilyPondDuration.LilyPondDuration.LilyPondDuration method), 2037

__ne__() (abjad.tools.lilypondparsertools.LilyPondEvent.LilyPondEvent.LilyPondEvent method), 2039

__ne__() (abjad.tools.lilypondparsertools.LilyPondFraction.LilyPondFraction.LilyPondFraction method), 2040

__ne__() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition.LilyPondLexicalDefinition method), 2044

__ne__() (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser method), 2048

<code>_ne__()</code>	<code>(abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject.MelodicIntervalClassObject)</code>	<code>method), 1083</code>	<code>method), 1201</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.HarmonicMelodicIntervalClassObject.MelodicIntervalClassObject)</code>	<code>method), 1085</code>	<code>method), 1101</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.HarmonicObject.HarmonicObject.HarmonicIntervalClassObject.MelodicIntervalObject.MelodicIntervalObject)</code>	<code>method), 1087</code>	<code>method), 1103</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment.IntervalClassObjectSegment)</code>	<code>method), 1088</code>	<code>method), 1104</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet)</code>	<code>method), 1091</code>	<code>method), 1206</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.IntervalObject.IntervalObject.IntervalObject)</code>	<code>method), 1093</code>	<code>method), 1208</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass.IntervalObjectClass)</code>	<code>method), 1094</code>	<code>method), 1211</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment.IntervalObjectSegment)</code>	<code>method), 1096</code>	<code>method), 1214</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet)</code>	<code>method), 1099</code>	<code>method), 1217</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.InversionEquivalentChromaticInterval(InversionEquivalentChromaticInterval))</code>	<code>method), 1158</code>	<code>method), 1220</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.InversionEquivalentChromaticInterval(InversionEquivalentChromaticInterval))</code>	<code>method), 1160</code>	<code>method), 1223</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.InversionEquivalentChromaticInterval(InversionEquivalentChromaticInterval))</code>	<code>method), 1163</code>	<code>method), 1227</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.InversionEquivalentChromaticInterval(InversionEquivalentChromaticInterval))</code>	<code>method), 1166</code>	<code>method), 1229</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.InversionEquivalentDiatonicInterval(InversionEquivalentDiatonicInterval))</code>	<code>method), 1168</code>	<code>method), 1232</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.InversionEquivalentDiatonicInterval(InversionEquivalentDiatonicInterval))</code>	<code>method), 1171</code>	<code>method), 1233</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.InversionEquivalentDiatonicInterval(InversionEquivalentDiatonicInterval))</code>	<code>method), 1174</code>	<code>method), 1235</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval)</code>	<code>method), 1176</code>	<code>method), 1238</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass)</code>	<code>method), 1177</code>	<code>method), 1242</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment)</code>	<code>method), 1179</code>	<code>method), 1246</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector)</code>	<code>method), 1182</code>	<code>method), 1248</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment)</code>	<code>method), 1185</code>	<code>method), 1250</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet)</code>	<code>method), 1188</code>	<code>method), 1105</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval)</code>	<code>method), 1190</code>	<code>method), 1107</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass)</code>	<code>method), 1192</code>	<code>method), 1108</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval)</code>	<code>method), 1194</code>	<code>method), 1110</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass)</code>	<code>method), 1196</code>	<code>method), 1113</code>
<code>_ne__()</code>	<code>(abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment)</code>	<code>method), 1199</code>	<code>method), 1116</code>

__ne__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner method), 1570	__ne__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner method), 1782
__ne__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1576	__ne__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1763
__ne__() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner method), 1584	__ne__() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner method), 1785
__ne__() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner method), 1590	__ne__() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner method), 1788
__ne__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1597	__ne__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1790
__ne__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner method), 1604	__ne__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner method), 1765
__ne__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner method), 1610	__ne__() (abjad.tools.timetokentools.TokenBurnishedSignalFilledTimeTokenMaker method), 1794
__ne__() (abjad.tools.spannertools.Spanner.Spanner.Spanner__ne__() (abjad.tools.spannertools.Spanner.Spanner method), 1511	__ne__() (abjad.tools.timetokentools.TokenIncisedNoteFilledTimeTokenMaker method), 1797
__ne__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner method), 1617	__ne__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner method), 1800
__ne__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner method), 1624	__ne__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner method), 1767
__ne__() (abjad.tools.spannertools.TextSpanner.TextSpanner method), 1630	__ne__() (abjad.tools.timetokentools.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker method), 1802
__ne__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner method), 1637	__ne__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1806
__ne__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff method), 1662	__ne__() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator method), 1809
__ne__() (abjad.tools.stafftools.Staff.Staff method), 1671	__ne__() (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator method), 1810
__ne__() (abjad.tools.tietools.TieChain.TieChain method), 1684	__ne__() (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator method), 1811
__ne__() (abjad.tools.tietools.TieSpanner.TieSpanner method), 1690	__ne__() (abjad.tools.tonalitytools.Mode.Mode.Mode method), 1813
__ne__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval method), 1709	__ne__() (abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator method), 1814
__ne__() (abjad.tools.timeintervaltools.TimeIntervalAggregator.Mixin method), 1703	__ne__() (abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator method), 1815
__ne__() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin method), 1706	__ne__() (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1818
__ne__() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1722	__ne__() (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree method), 1820
__ne__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1744	__ne__() (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator method), 1821
__ne__() (abjad.tools.timetokentools.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker method), 1759	__ne__() (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction method), 1822
__ne__() (abjad.tools.timetokentools.EqualDivisionTimeTokenMaker.EqualDivisionTimeTokenMaker method), 1770	__ne__() (abjad.tools.tonalitytools.TupleTupletMaker.TupleTupletMaker method), 1838
__ne__() (abjad.tools.timetokentools.IncisedTimeTokenMaker.IncisedTimeTokenMaker method), 1761	__ne__() (abjad.tools.tonalitytools.TupleTupletMaker.TupleTupletMaker method), 1848
__ne__() (abjad.tools.timetokentools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker method), 1773	__ne__() (abjad.tools.tonalitytools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker method), 1861
__ne__() (abjad.tools.timetokentools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker method), 1776	__ne__() (abjad.tools.tonalitytools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker method), 1873
__ne__() (abjad.tools.timetokentools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker method), 1779	__ne__() (abjad.tools.tonalitytools.QuantifiedNoteCheckBeats method), 2083

`__ne__()` (abjad.tools.wellformednesstools.Check.Check.CheckNonzero method), 2081
`__ne__()` (abjad.tools.wellformednesstools.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck method), 2084
`__ne__()` (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck method), 2086
`__ne__()` (abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck method), 2087
`__ne__()` (abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck method), 2089
`__ne__()` (abjad.tools.wellformednesstools.MisduratedMeasureCheck.MisduratedMeasureCheck method), 2090
`__ne__()` (abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck method), 2092
`__ne__()` (abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck method), 2093
`__ne__()` (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck method), 2095
`__ne__()` (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck method), 2096
`__ne__()` (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck method), 2098
`__ne__()` (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck method), 2099
`__ne__()` (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck method), 2101
`__ne__()` (abjad.tools.wellformednesstools.OverlappingOctaveCheck.OverlappingOctaveCheck method), 2102
`__ne__()` (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck method), 2104
`__neg__()` (abjad.tools.durationtools.Duration.Duration.Duration method), 447
`__neg__()` (abjad.tools.durationtools.Offset.Offset.Offset method), 451
`__neg__()` (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction method), 951
`__neg__()` (abjad.tools.pitchtools.Accidental.Accidental.Accidental method), 1130
`__neg__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1158
`__neg__()` (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval method), 1176
`__neg__()` (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval method), 1190
`__neg__()` (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1194
`__neg__()` (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject method), 1103
`__neg__()` (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch method), 1232
`__neg__()` (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1234
`__nonzero__()` (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark method), 421
`__nonzero__()` (abjad.tools.durationtools.Duration.Duration.Duration method), 447
`__nonzero__()` (abjad.tools.durationtools.Offset.Offset.Offset method), 451
`__nonzero__()` (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction method), 951
`__nonzero__()` (abjad.tools.pitchtools.Accidental.Accidental.Accidental method), 1130
`__nonzero__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1163
`__nonzero__()` (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval method), 1201
`__nonzero__()` (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval method), 1214
`__nonzero__()` (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1220
`__nonzero__()` (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject method), 1242
`__nonzero__()` (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch method), 1113
`__nonzero__()` (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1122
`__nonzero__()` (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark method), 1128

Index	2177
--------------	-------------

__repr__() (abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.AContextMark.AContextMark.AContextMark method), 1877

__repr__() (abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject method), 1878

__repr__() (abjad.tools.abctools.Parser.Parser.Parser __repr__() (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark method), 1880

__repr__() (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection.ScoreSelection.ScoreSelection method), 1882

__repr__() (abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject method), 1883

__repr__() (abjad.tools.abjadbooktools.AbjadBookProcessor.AbjadBookProcessor.AbjadBookProcessor.AbjadBookProcessor.AbjadBookProcessor method), 1886

__repr__() (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.AbjadBookScript.AbjadBookScript.AbjadBookScript method), 1888

__repr__() (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock.CodeBlock.CodeBlock method), 1889

__repr__() (abjad.tools.abjadbooktools.HTMLOutputFormat.HTMLOutputFormat.HTMLOutputFormat.HTMLOutputFormat.HTMLOutputFormat method), 1891

__repr__() (abjad.tools.abjadbooktools.LaTeXOutputFormat.LaTeXOutputFormat.LaTeXOutputFormat.LaTeXOutputFormat.LaTeXOutputFormat method), 1893

__repr__() (abjad.tools.abjadbooktools.OutputFormat.OutputFormat.OutputFormat.OutputFormat.OutputFormat method), 1884

__repr__() (abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutputFormat.ReSTOutputFormat.ReSTOutputFormat.ReSTOutputFormat method), 1894

__repr__() (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner.BeamSpanner.BeamSpanner method), 229

__repr__() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 237

__repr__() (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 246

__repr__() (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 255

__repr__() (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 261

__repr__() (abjad.tools.chordtools.Chord.Chord.Chord __repr__() (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScript method), 271

__repr__() (abjad.tools.componenttools.Component.Component.Component.Component.Component method), 277

__repr__() (abjad.tools.componenttools.ContainmentSignatures.ContainmentSignatures.ContainmentSignatures.ContainmentSignatures.ContainmentSignatures method), 279

__repr__() (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig.AbjadConfig.AbjadConfig method), 1897

__repr__() (abjad.tools.containertools.Cluster.Cluster.Cluster __repr__() (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript method), 341

__repr__() (abjad.tools.containertools.Container.Container.Container __repr__() (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript.RenameModulesScript method), 348

__repr__() (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer method), 356

__repr__() (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark __repr__() (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript method), 380

__repr__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 383

__repr__() (abjad.tools.contexttools.Context.Context.Context __repr__() (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScript method), 391

__repr__() (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript method), 1946

__repr__() (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript.ExtraSpannerError.ExtraSpannerError method), 1948

__repr__() (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript.SvnMessageScript.GraceContainerError.GraceContainerError method), 1950

__repr__() (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript.ImpreciseTempoError.ImpreciseTempoError method), 1953

__repr__() (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler.InputSpecificationError.InputSpecificationError method), 1954

__repr__() (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator.InstrumentError.InstrumentError method), 1956

__repr__() (abjad.tools.documentationtools.ClassCrawler.ClassCrawler(ClassCrawler.exceptiontools.IntervalError.IntervalError method), 1958

__repr__() (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter(ClassDocumenter.LilyPondParserError.LilyPondParserError method), 1960

__repr__() (abjad.tools.documentationtools.Documenter.Documenter(Documenter.exceptiontools.LineBreakError.LineBreakError method), 1961

__repr__() (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler(FunctionCrawler.MarkError.MarkError method), 1963

__repr__() (abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter(FunctionDocumenter.MeasureContiguityError.MeasureContiguityError method), 1965

__repr__() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph(InheritanceGraph.MeasureError.MeasureError method), 1968

__repr__() (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler(ModuleCrawler.MissingComponentError.MissingComponentError method), 1969

__repr__() (abjad.tools.documentationtools.Pipe.Pipe.Pipe __repr__() (abjad.tools.exceptiontools.MissingInstrumentError.MissingInstrumentError method), 1972

__repr__() (abjad.tools.durationtools.Duration.Duration.Duration __repr__() (abjad.tools.exceptiontools.MissingMarkError.MissingMarkError method), 447

__repr__() (abjad.tools.durationtools.Offset.Offset.Offset __repr__() (abjad.tools.exceptiontools.MissingMeasureError.MissingMeasureError method), 452

__repr__() (abjad.tools.durationtools.TimespanConstant.TimespanConstant.TimespanConstant __repr__() (abjad.tools.exceptiontools.MissingNamedComponentError.MissingNamedComponentError method), 454

__repr__() (abjad.tools.exceptiontools.AssignabilityError.AssignabilityError __repr__() (abjad.tools.exceptiontools.MissingNoteHeadError.MissingNoteHeadError method), 1974

__repr__() (abjad.tools.exceptiontools.ClefError.ClefError __repr__() (abjad.tools.exceptiontools.MissingPitchError.MissingPitchError method), 1975

__repr__() (abjad.tools.exceptiontools.ContainmentError.ContainmentError __repr__() (abjad.tools.exceptiontools.MissingSpannerError.MissingSpannerError method), 1976

__repr__() (abjad.tools.exceptiontools.ContextContainmentError.ContextContainmentError __repr__() (abjad.tools.exceptiontools.MissingTempoError.MissingTempoError method), 1977

__repr__() (abjad.tools.exceptiontools.ContiguityError.ContiguityError __repr__() (abjad.tools.exceptiontools.MusicContentsError.MusicContentsError method), 1978

__repr__() (abjad.tools.exceptiontools.CyclicNodeError.CyclicNodeError __repr__() (abjad.tools.exceptiontools.NegativeDurationError.NegativeDurationError method), 1979

__repr__() (abjad.tools.exceptiontools.DurationError.DurationError __repr__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError method), 1980

__repr__() (abjad.tools.exceptiontools.ExtraMarkError.ExtraMarkError __repr__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError method), 1981

__repr__() (abjad.tools.exceptiontools.ExtraNamedComponentError.ExtraNamedComponentError __repr__() (abjad.tools.exceptiontools.NoteHeadError.NoteHeadError method), 1982

__repr__() (abjad.tools.exceptiontools.ExtraNoteHeadError.ExtraNoteHeadError __repr__() (abjad.tools.exceptiontools.OverfullContainerError.OverfullContainerError method), 1983

__repr__() (abjad.tools.exceptiontools.ExtraPitchError.ExtraPitchError __repr__() (abjad.tools.exceptiontools.ParallelError.ParallelError method), 1984

[__repr__\(\) \(abjad.tools.exceptiontools.PartitionError.PartitionError method\), 1012](#)
[__repr__\(\) \(abjad.tools.exceptiontools.PitchError.PitchError method\), 1013](#)
[__repr__\(\) \(abjad.tools.exceptiontools.SchemeParserFinishedException.SchemeParserFinishedException method\), 1014](#)
[__repr__\(\) \(abjad.tools.exceptiontools.SpacingError.SpacingError method\), 1015](#)
[__repr__\(\) \(abjad.tools.exceptiontools.SpannerError.SpannerError method\), 1016](#)
[__repr__\(\) \(abjad.tools.exceptiontools.SpannerPopulationError.SpannerPopulationError method\), 1017](#)
[__repr__\(\) \(abjad.tools.exceptiontools.StaffContainmentError.StaffContainmentError method\), 1018](#)
[__repr__\(\) \(abjad.tools.exceptiontools.TempoError.TempoError method\), 1019](#)
[__repr__\(\) \(abjad.tools.exceptiontools.TieChainError.TieChainError method\), 1020](#)
[__repr__\(\) \(abjad.tools.exceptiontools.TimeSignatureAssignmentError.TimeSignatureAssignmentError method\), 1021](#)
[__repr__\(\) \(abjad.tools.exceptiontools.TimeSignatureError.TimeSignatureError method\), 1022](#)
[__repr__\(\) \(abjad.tools.exceptiontools.TonalHarmonyError.TonalHarmonyError method\), 1023](#)
[__repr__\(\) \(abjad.tools.exceptiontools.TupletError.TupletError method\), 1024](#)
[__repr__\(\) \(abjad.tools.exceptiontools.TupletFuseError.TupletFuseError method\), 1025](#)
[__repr__\(\) \(abjad.tools.exceptiontools.TypographicWhitespaceError.TypographicWhitespaceError method\), 1026](#)
[__repr__\(\) \(abjad.tools.exceptiontools.UnboundedTimeIntervalError.UnboundedTimeIntervalError method\), 1027](#)
[__repr__\(\) \(abjad.tools.exceptiontools.UndefinedSpacingError.UndefinedSpacingError method\), 1028](#)
[__repr__\(\) \(abjad.tools.exceptiontools.UnderfullContainerError.UnderfullContainerError method\), 1029](#)
[__repr__\(\) \(abjad.tools.exceptiontools.VoiceContainmentError.VoiceContainmentError method\), 1030](#)
[__repr__\(\) \(abjad.tools.gracetools.GraceContainer.GraceContainer method\), 482](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Accordion.Accordion method\), 491](#)
[__repr__\(\) \(abjad.tools.instrumenttools.AltoFlute.AltoFlute method\), 497](#)
[__repr__\(\) \(abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone method\), 503](#)
[__repr__\(\) \(abjad.tools.instrumenttools.AltoTrombone.AltoTrombone method\), 509](#)
[__repr__\(\) \(abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet method\), 515](#)
[__repr__\(\) \(abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone method\), 521](#)
[__repr__\(\) \(abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice method\), 527](#)
[__repr__\(\) \(abjad.tools.instrumenttools.BassClarinet.BassClarinet method\), 533](#)
[__repr__\(\) \(abjad.tools.instrumenttools.BassFlute.BassFlute method\), 539](#)
[__repr__\(\) \(abjad.tools.instrumenttools.BassSaxophone.BassSaxophone method\), 545](#)
[__repr__\(\) \(abjad.tools.instrumenttools.BassTrombone.BassTrombone method\), 551](#)
[__repr__\(\) \(abjad.tools.instrumenttools.BassVoice.BassVoice method\), 557](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Bassoon.Bassoon method\), 563](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Cello.Cello method\), 569](#)
[__repr__\(\) \(abjad.tools.instrumenttools.ClarinetInA.ClarinetInA method\), 575](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Contrabass.Contrabass method\), 581](#)
[__repr__\(\) \(abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet method\), 587](#)
[__repr__\(\) \(abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute method\), 593](#)
[__repr__\(\) \(abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone method\), 599](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Contrabassoon.Contrabassoon method\), 605](#)
[__repr__\(\) \(abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice method\), 611](#)
[__repr__\(\) \(abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet method\), 617](#)
[__repr__\(\) \(abjad.tools.instrumenttools.EnglishHorn.EnglishHorn method\), 623](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Flute.Flute method\), 629](#)
[__repr__\(\) \(abjad.tools.instrumenttools.FrenchHorn.FrenchHorn method\), 634](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Glockenspiel.Glockenspiel method\), 640](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Guitar.Guitar method\), 646](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Harp.Harp method\), 652](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Harpsichord.Harpsichord method\), 658](#)
[__repr__\(\) \(abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory method\), 661](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Marimba.Marimba method\), 667](#)
[__repr__\(\) \(abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice method\), 673](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Oboe.Oboe method\), 679](#)
[__repr__\(\) \(abjad.tools.instrumenttools.Piano.Piano method\), 685](#)

__repr__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 691

__repr__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 697

__repr__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 703

__repr__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 709

__repr__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 715

__repr__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 721

__repr__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 727

__repr__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 733

__repr__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 739

__repr__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 745

__repr__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 751

__repr__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 757

__repr__() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 763

__repr__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 769

__repr__() (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 825

__repr__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 830

__repr__() (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 862

__repr__() (abjad.tools.lilypondfiletools.AttributedBlock.AttributedBlock.AttributedBlock method), 858

__repr__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 865

__repr__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 868

__repr__() (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock method), 870

__repr__() (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken method), 871

__repr__() (abjad.tools.lilypondfiletools.HeaderBlock.HeaderBlock.HeaderBlock method), 872

__repr__() (abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock.LayoutBlock method), 874

__repr__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 877

__repr__() (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 879

__repr__() (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken method), 881

__repr__() (abjad.tools.lilypondfiletools.MIDIBlock.MIDIBlock.MIDIBlock method), 882

__repr__() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 861

__repr__() (abjad.tools.lilypondfiletools.PaperBlock.PaperBlock.PaperBlock method), 883

__repr__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 886

__repr__() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy method), 2036

__repr__() (abjad.tools.lilypondparsertools.LilyPondDuration.LilyPondDuration.LilyPondDuration method), 2037

__repr__() (abjad.tools.lilypondparsertools.LilyPondEvent.LilyPondEvent.LilyPondEvent method), 2039

__repr__() (abjad.tools.lilypondparsertools.LilyPondFraction.LilyPondFraction.LilyPondFraction method), 2040

__repr__() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition.LilyPondLexicalDefinition method), 2044

__repr__() (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser method), 2048

__repr__() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2067

__repr__() (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser method), 2073

__repr__() (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser method), 2078

__repr__() (abjad.tools.lilypondparsertools.SyntaxNode.SyntaxNode.SyntaxNode method), 2079

__repr__() (abjad.tools.marktools.Annotation.Annotation.Annotation method), 893

__repr__() (abjad.tools.marktools.Articulation.Articulation.Articulation method), 895

__repr__() (abjad.tools.marktools.BarLine.BarLine.BarLine method), 899

__repr__() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 902

__repr__() (abjad.tools.marktools.DirectedMark.DirectedMark.DirectedMark method), 890

__repr__() (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 905

__repr__() (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method), 908

__repr__() (abjad.tools.marktools.Mark.Mark.Mark method), 910

__repr__() (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method), 913

__repr__() (abjad.tools.markuptools.Markup.Markup.Markup method), 934

__repr__() (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand method), 936

__repr__() (abjad.tools.markuptools.MultiphaseMarkupInventory.MultiphaseMarkupInventory.MultiphaseMarkupInventory method), 939

__repr__() (abjad.tools.marktools.BoundedObject.BoundedObject.BoundedObject method), 946

__repr__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.__repr__ method), 952

__repr__() (abjad.tools.mathtools.Ratio.Ratio.Ratio.__repr__ method), 955

__repr__() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.__repr__ method), 988

__repr__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.__repr__ method), 999

__repr__() (abjad.tools.measuretools.Measure.Measure.Measure.__repr__ method), 1009

__repr__() (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.__repr__ method), 1036

__repr__() (abjad.tools.notetools.Note.Note.Note.__repr__ method), 1040

__repr__() (abjad.tools.notetools.NoteHead.NoteHead.NoteHead.__repr__ method), 1043

__repr__() (abjad.tools.pitcharraytools.PitchArray.PitchArray.__repr__ method), 1051

__repr__() (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.__repr__ method), 1054

__repr__() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.__repr__ method), 1056

__repr__() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.__repr__ method), 1058

__repr__() (abjad.tools.pitchtools.Accidental.Accidental.Accidental.__repr__ method), 1130

__repr__() (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject.__repr__ method), 1065

__repr__() (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.__repr__ method), 1067

__repr__() (abjad.tools.pitchtools.ChromaticObject.ChromaticObject.__repr__ method), 1068

__repr__() (abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject.__repr__ method), 1070

__repr__() (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject.__repr__ method), 1071

__repr__() (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.__repr__ method), 1073

__repr__() (abjad.tools.pitchtools.CounterpointObject.CounterpointObject.__repr__ method), 1074

__repr__() (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject.__repr__ method), 1076

__repr__() (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.__repr__ method), 1077

__repr__() (abjad.tools.pitchtools.DiatonicObject.DiatonicObject.__repr__ method), 1079

__repr__() (abjad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClassObject.__repr__ method), 1080

__repr__() (abjad.tools.pitchtools.DiatonicPitchObject.DiatonicPitchObject.__repr__ method), 1082

__repr__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassObject.HarmonicChromaticIntervalClassObject.__repr__ method), 1132

__repr__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.__repr__ method), 1133

__repr__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.__repr__ method), 1137

__repr__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.__repr__ method), 1139

__repr__() (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.__repr__ method), 1141

__repr__() (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.__repr__ method), 1143

__repr__() (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.__repr__ method), 1145

__repr__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.__repr__ method), 1146

__repr__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.__repr__ method), 1148

__repr__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.__repr__ method), 1151

__repr__() (abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject.__repr__ method), 1154

__repr__() (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.__repr__ method), 1156

__repr__() (abjad.tools.pitchtools.HarmonicIntervalClassSet.HarmonicIntervalClassSet.__repr__ method), 1183

__repr__() (abjad.tools.pitchtools.HarmonicIntervalObjectSet.HarmonicIntervalObjectSet.__repr__ method), 1085

__repr__() (abjad.tools.pitchtools.HarmonicObject.HarmonicObject.__repr__ method), 1087

__repr__() (abjad.tools.pitchtools.HarmonicObjectClass.HarmonicObjectClass.__repr__ method), 1089

__repr__() (abjad.tools.pitchtools.HarmonicObjectSet.HarmonicObjectSet.__repr__ method), 1091

__repr__() (abjad.tools.pitchtools.IntervalClassObject.IntervalClassObject.__repr__ method), 1093

__repr__() (abjad.tools.pitchtools.IntervalObject.IntervalObject.__repr__ method), 1094

__repr__() (abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass.__repr__ method), 1096

__repr__() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.__repr__ method), 1099

__repr__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassObject.InversionEquivalentChromaticIntervalClassObject.__repr__ method), 1158

__repr__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.__repr__ method), 1161

__repr__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalObject.InversionEquivalentChromaticIntervalObject.__repr__ method), 1163

__repr__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassObject.InversionEquivalentDiatonicIntervalClassObject.__repr__ method), 1166

__repr__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.__repr__ method), 1168

__repr__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalObject.InversionEquivalentDiatonicIntervalObject.__repr__ method), 1171

__repr__() (abjad.tools.pitchtools.InversionEquivalentHarmonicChromaticIntervalClassObject.InversionEquivalentHarmonicChromaticIntervalClassObject.__repr__ method), 1174

__repr__() (abjad.tools.pitchtools.InversionEquivalentHarmonicChromaticIntervalClass.InversionEquivalentHarmonicChromaticIntervalClass.__repr__ method), 1176

__repr__() (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser.ResidueClassExpression.ResidueClassExpression.
method), 1346

__repr__() (abjad.tools.schemetools.Scheme.Scheme.Scheme.__repr__() (abjad.tools.skiptools.Skip.Skip.Skip.
method), 1348

__repr__() (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList.BracketSpanner.BracketSpanner.BracketSpanner.
method), 1350

__repr__() (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner.
method), 1352

__repr__() (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner.
method), 1353

__repr__() (abjad.tools.schemetools.SchemePair.SchemePair.SchemePair.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner.
method), 1354

__repr__() (abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector.DirectedSpanner.DirectedSpanner.DirectedSpanner.
method), 1356

__repr__() (abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant.SchemeVectorConstant.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner.
method), 1358

__repr__() (abjad.tools.scoretemplatetools.GroupedRhythmicStaffScoreTemplate.GroupedRhythmicStaffScoreTemplate.GroupedRhythmicStaffScoreTemplate.
method), 1362

__repr__() (abjad.tools.scoretemplatetools.GroupedStavesScoreTemplate.GroupedStavesScoreTemplate.GroupedStavesScoreTemplate.
method), 1364

__repr__() (abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplate.ScoreTemplate.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner.
method), 1360

__repr__() (abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.StringQuartetScoreTemplate.StringQuartetScoreTemplate.HorizontalE
method), 1366

__repr__() (abjad.tools.scoretemplatetools.TwoStaffPianoScoreTemplate.TwoStaffPianoScoreTemplate.TwoStaffPianoScoreTemplate.
method), 1368

__repr__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner.
method), 1376

__repr__() (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier.InstrumentationSpecifier.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner.
method), 1379

__repr__() (abjad.tools.scoretools.Performer.Performer.Performer() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner.
method), 1382

__repr__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory.SlurSpanner.SlurSpanner.SlurSpanner.
method), 1384

__repr__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff() (abjad.tools.spannertools.Spanner.Spanner.Spanner.
method), 1393

__repr__() (abjad.tools.scoretools.Score.Score.Score __repr__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner.
method), 1402

__repr__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner.
method), 1411

__repr__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner.
method), 1419

__repr__() (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix.CyclicMatrix() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner.
method), 1422

__repr__() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff.
method), 1431

__repr__() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple() (abjad.tools.stafftools.Staff.Staff.Staff.
method), 1433

__repr__() (abjad.tools.sequencetools.Matrix.Matrix.Matrix __repr__() (abjad.tools.tietools.TieChain.TieChain.TieChain.
method), 1435

__repr__() (abjad.tools.sequencetools.Tree.Tree.Tree __repr__() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner.
method), 1444

__repr__() (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval.
method), 1492

__repr__() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TahjalToolsAggregateMQualTiyIndicatvalAggityIndMaxon Qual
method), 1703 method), 1815

__repr__() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMihjadToolsIntervalMixols.Scale.Scale.Scale
method), 1706 method), 1818

__repr__() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeInter(OAbjadToolsIntervalTTools.ScaleDegree.ScaleDegree.ScaleDegree
method), 1722 method), 1820

__repr__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary(TahjalToolsIntervalTTools.SuspensionIndicat
method), 1744 method), 1821

__repr__() (abjad.tools.timetoken tools.BurnishedTimeTokenMaker.BornishadToolsTokenMakers.BornishadTimeTokenMaker.TonalFun
method), 1759 method), 1822

__repr__() (abjad.tools.timetoken tools.EqualDivisionTimeTokenMaker(EbjadToolsTimeTokenMakers.EqualDivisionTimeTokenMaker
method), 1770 method), 1839

__repr__() (abjad.tools.timetoken tools.IncisedTimeTokenMaker.IncisedTimeTokenMaker.IncisedTimeTokenMaker.TupleTuplet
method), 1761 method), 1848

__repr__() (abjad.tools.timetoken tools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.Moment.Ve
method), 1773 method), 1861

__repr__() (abjad.tools.timetoken tools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker.C
method), 1776 method), 1873

__repr__() (abjad.tools.timetoken tools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker.NoteOutp
method), 1779 method), 2083

__repr__() (abjad.tools.timetoken tools.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker.OutputInci
method), 1782 method), 2081

__repr__() (abjad.tools.timetoken tools.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker
method), 1763 method), 2084

__repr__() (abjad.tools.timetoken tools.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.DuplicateId
method), 1785 method), 2086

__repr__() (abjad.tools.timetoken tools.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.Empty
method), 1788 method), 2087

__repr__() (abjad.tools.timetoken tools.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker.Check.Inte
method), 1790 method), 2089

__repr__() (abjad.tools.timetoken tools.TimeTokenMaker.TimeTokenMaker.TimeTokenMaker.MisduatedMeasureCheck.Mis
method), 1765 method), 2090

__repr__() (abjad.tools.timetoken tools.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker.Mis
method), 1794 method), 2092

__repr__() (abjad.tools.timetoken tools.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker.TokenIncise
method), 1797 method), 2093

__repr__() (abjad.tools.timetoken tools.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker.TokenIncise
method), 1800 method), 2095

__repr__() (abjad.tools.timetoken tools.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.Mis
method), 1767 method), 2096

__repr__() (abjad.tools.timetoken tools.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker.NestedM
method), 1802 method), 2098

__repr__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass(abjad.tools.wellformednesstools.OverlappingBeamCheck.Overl
method), 1806 method), 2099

__repr__() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator(abjad.tools.wellformednesstools.OverlappingGlissandoCheck.C
method), 1809 method), 2101

__repr__() (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator(abjad.tools.wellformednesstools.OverlappingOctavationCheck.O
method), 1810 method), 2102

__repr__() (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator(abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairp
method), 1811 method), 2104

__repr__() (abjad.tools.tonalitytools.Mode.Mode.Mode __reversed__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInven
method), 1813 method), 383

__repr__() (abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator(abjad.tools.contexttools.TempoMarkInventory.TempoMark
method), 1814 method), 417

`__reversed__()` (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory) (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory) (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory) (method), 1907
`__reversed__()` (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory) (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory) (method), 661
`__reversed__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock) (abjad.tools.durationtools.Offset.Offset) (method), 865
`__reversed__()` (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock) (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock) (method), 868
`__reversed__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile) (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile) (method), 877
`__reversed__()` (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock) (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock) (method), 861
`__reversed__()` (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock) (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock) (method), 886
`__reversed__()` (abjad.tools.markuptools.MarkupInventory.MarkupInventory) (abjad.tools.markuptools.MarkupInventory.MarkupInventory) (method), 939
`__reversed__()` (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping) (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping) (method), 1253
`__reversed__()` (abjad.tools.pitchtools.OctaveTranspositionMapping.InvertedOctaveTranspositionMapping) (abjad.tools.pitchtools.OctaveTranspositionMapping.InvertedOctaveTranspositionMapping) (method), 1259
`__reversed__()` (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory) (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory) (method), 1264
`__reversed__()` (abjad.tools.scoretools.PerformerInventory.PerformerInventory) (abjad.tools.scoretools.PerformerInventory.PerformerInventory) (method), 1384
`__reversed__()` (abjad.tools.sequencetools.CyclicList.CyclicList) (abjad.tools.sequencetools.CyclicList.CyclicList) (method), 1419
`__rfloordiv__()` (abjad.tools.durationtools.Duration.Duration) (abjad.tools.mathtools.Ratio.Ratio) (method), 447
`__rfloordiv__()` (abjad.tools.durationtools.Offset.Offset) (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure) (method), 452
`__rfloordiv__()` (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction) (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure) (method), 952
`__rmod__()` (abjad.tools.durationtools.Duration.Duration) (abjad.tools.measuretools.Measure.Measure) (method), 448
`__rmod__()` (abjad.tools.durationtools.Offset.Offset) (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic) (method), 452
`__rmod__()` (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction) (abjad.tools.notetools.Note.Note) (method), 952
`__rmul__()` (abjad.tools.chordtools.Chord.Chord) (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment) (method), 271
`__rmul__()` (abjad.tools.componenttools.Component.Component) (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment) (method), 277
`__rmul__()` (abjad.tools.containertools.Cluster.Cluster) (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment) (method), 341
`__rmul__()` (abjad.tools.containertools.Container.Container) (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment) (method), 348
`__rmul__()` (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer) (abjad.tools.pitchtools.IntervalChromaticIntervalClassSegment.IntervalChromaticIntervalClassSegment) (method), 356
`__rmul__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory) (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment) (method), 383
`__rmul__()` (abjad.tools.contexttools.Context.Context) (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment) (method), 392
`__rmul__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory) (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment) (method), 417

__mul__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1195

__mul__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 1199

__mul__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1211

__mul__() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment method), 1217

__mul__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1238

__mul__() (abjad.tools.pitchtools.ObjectSegment.ObjectSegment method), 1110

__mul__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1253

__mul__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1259

__mul__() (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment method), 1119

__mul__() (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment method), 1126

__mul__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory method), 1265

__mul__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow method), 1268

__mul__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest method), 1310

__mul__() (abjad.tools.resttools.Rest.Rest.Rest method), 1313

__mul__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1376

__mul__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory method), 1384

__mul__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1393

__mul__() (abjad.tools.scoretools.Score.Score.Score method), 1402

__mul__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1411

__mul__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1419

__mul__() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple method), 1433

__mul__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1497

__mul__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1662

__mul__() (abjad.tools.stafftools.Staff.Staff.Staff method), 1671

__mul__() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator method), 1809

__mul__() (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1818

__mul__() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet method), 1839

__mul__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1848

__mul__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 1873

__mul__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1141

__mul__() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment method), 1151

__mul__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1156

__mul__() (abjad.tools.pitchtools.ObjectSegment.ObjectSegment method), 1091

__mul__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1099

__mul__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1163

__mul__() (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment method), 1188

__mul__() (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment method), 1201

__mul__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory method), 1214

__mul__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow method), 1220

__mul__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest method), 1242

__mul__() (abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet method), 1113

__mul__() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1122

__mul__() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet method), 1128

__mul__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1806

__mul__() (abjad.tools.durationtools.Duration.Duration.Duration method), 448

__mul__() (abjad.tools.durationtools.Offset.Offset.Offset method), 452

__mul__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction method), 952

__mul__() (abjad.tools.durationtools.Duration.Duration.Duration method), 448

__mul__() (abjad.tools.durationtools.Offset.Offset.Offset method), 452

__mul__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction method), 952

__mul__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1142

__mul__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1151

__mul__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1157

__mul__() (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet method), 1091

__rsub__() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet.PitchObjectSet.PitchObjectSet.PitchObjectSet.PitchObjectSet method), 1099

__rsub__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.IntervalClassSet.IntervalClassSet.IntervalClassSet.IntervalClassSet.IntervalClassSet method), 1163

__rsub__() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1188

__rsub__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1202

__rsub__() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1214

__rsub__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1220

__rsub__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1242

__rsub__() (abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet.__setattr__() (abjad.tools.exceptiontools.CyclicNodeError.CyclicNodeError.CyclicNodeError.CyclicNodeError.CyclicNodeError method), 1113

__rsub__() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1122

__rsub__() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet.PitchObjectSet.PitchObjectSet method), 1128

__rsub__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.ChordClass.ChordClass method), 1806

__rtruediv__() (abjad.tools.durationtools.Duration.Duration.Duration.Duration.Duration method), 448

__rtruediv__() (abjad.tools.durationtools.Offset.Offset.Offset.__setattr__() (abjad.tools.exceptiontools.ExtraPitchError.ExtraPitchError.ExtraPitchError.ExtraPitchError.ExtraPitchError method), 452

__rtruediv__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction.NonreducedFraction.NonreducedFraction method), 952

__rxor__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1142

__rxor__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1151

__rxor__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1157

__rxor__() (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet method), 1091

__rxor__() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet method), 1099

__rxor__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.IntervalClassSet.IntervalClassSet.IntervalClassSet.IntervalClassSet method), 1163

__rxor__() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1188

__rxor__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1202

__rxor__() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1214

__rxor__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1220

__rxor__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1243

__rxor__() (abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet.__setattr__() (abjad.tools.exceptiontools.MissingInstrumentError.MissingInstrumentError.MissingInstrumentError.MissingInstrumentError.MissingInstrumentError method), 1113

__rxor__() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1122

<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingMeasureError.MissingMeasureError.MissingMeasureError method), 1999	<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingMeasureError.MissingMeasureError.MissingMeasureError method), 2026
<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingNamedComponentError.MissingNamedComponentError.MissingNamedComponentError method), 2000	<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingNamedComponentError.MissingNamedComponentError.MissingNamedComponentError method), 2027
<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingNoteHeadError.MissingNoteHeadError.MissingNoteHeadError method), 2001	<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingNoteHeadError.MissingNoteHeadError.MissingNoteHeadError method), 2028
<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingPitchError.MissingPitchError.MissingPitchError method), 2002	<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingPitchError.MissingPitchError.MissingPitchError method), 2029
<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingSpannerError.MissingSpannerError.MissingSpannerError method), 2003	<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingSpannerError.MissingSpannerError.MissingSpannerError method), 2030
<code>__setattr__()</code> (abjad.tools.exceptiontools.MissingTempoError.MissingTempoError.MissingTempoError method), 2004	<code>__setattr__()</code> (abjad.tools.chordtools.Chord.Chord.Chord method), 271
<code>__setattr__()</code> (abjad.tools.exceptiontools.MusicContentsError.MusicContentsError.MusicContentsError method), 2005	<code>__setattr__()</code> (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1897
<code>__setattr__()</code> (abjad.tools.exceptiontools.NegativeDurationError.NegativeDurationError.NegativeDurationError method), 2006	<code>__setattr__()</code> (abjad.tools.containertools.Cluster.Cluster.Cluster method), 342
<code>__setattr__()</code> (abjad.tools.exceptiontools.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError method), 2007	<code>__setattr__()</code> (abjad.tools.containertools.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError method), 348
<code>__setattr__()</code> (abjad.tools.exceptiontools.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError method), 2008	<code>__setattr__()</code> (abjad.tools.containertools.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError method), 356
<code>__setattr__()</code> (abjad.tools.exceptiontools.NoteHeadError.NoteHeadError.NoteHeadError method), 2009	<code>__setattr__()</code> (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 383
<code>__setattr__()</code> (abjad.tools.exceptiontools.OverfullContainerError.OverfullContainerError.OverfullContainerError method), 2010	<code>__setattr__()</code> (abjad.tools.contexttools.Context.Context.Context method), 392
<code>__setattr__()</code> (abjad.tools.exceptiontools.ParallelError.ParallelError.ParallelError method), 2011	<code>__setattr__()</code> (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 417
<code>__setattr__()</code> (abjad.tools.exceptiontools.PartitionError.PartitionError.PartitionError method), 2012	<code>__setattr__()</code> (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method), 1904
<code>__setattr__()</code> (abjad.tools.exceptiontools.PitchError.PitchError.PitchError method), 2013	<code>__setattr__()</code> (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1907
<code>__setattr__()</code> (abjad.tools.exceptiontools.SchemeParserFinishedException.SchemeParserFinishedException.SchemeParserFinishedException method), 2014	<code>__setattr__()</code> (abjad.tools.inheritance.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1968
<code>__setattr__()</code> (abjad.tools.exceptiontools.SpacingError.SpacingError.SpacingError method), 2015	<code>__setattr__()</code> (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 482
<code>__setattr__()</code> (abjad.tools.exceptiontools.SpannerError.SpannerError.SpannerError method), 2016	<code>__setattr__()</code> (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 661
<code>__setattr__()</code> (abjad.tools.exceptiontools.SpannerPopulationError.SpannerPopulationError.SpannerPopulationError method), 2017	<code>__setattr__()</code> (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 865
<code>__setattr__()</code> (abjad.tools.exceptiontools.StaffContainmentError.StaffContainmentError.StaffContainmentError method), 2018	<code>__setattr__()</code> (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 868
<code>__setattr__()</code> (abjad.tools.exceptiontools.TempoError.TempoError.TempoError method), 2019	<code>__setattr__()</code> (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 878
<code>__setattr__()</code> (abjad.tools.exceptiontools.TieChainError.TieChainError.TieChainError method), 2020	<code>__setattr__()</code> (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 861
<code>__setattr__()</code> (abjad.tools.exceptiontools.TimeSignatureAssignmentError.TimeSignatureAssignmentError.TimeSignatureAssignmentError method), 2021	<code>__setattr__()</code> (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 886
<code>__setattr__()</code> (abjad.tools.exceptiontools.TimeSignatureError.TimeSignatureError.TimeSignatureError method), 2022	<code>__setattr__()</code> (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 939
<code>__setattr__()</code> (abjad.tools.exceptiontools.TonalHarmonyError.TonalHarmonyError.TonalHarmonyError method), 2023	<code>__setattr__()</code> (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 988
<code>__setattr__()</code> (abjad.tools.exceptiontools.TupletError.TupletError.TupletError method), 2024	<code>__setattr__()</code> (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 999
<code>__setattr__()</code> (abjad.tools.exceptiontools.TupletFuseError.TupletFuseError.TupletFuseError method), 2025	<code>__setattr__()</code> (abjad.tools.measuretools.Measure.Measure.Measure method), 1009

__getitem__() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method), 1051

__getitem__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1137

__getitem__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector method), 1167

__getitem__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector method), 1174

__getitem__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 1182

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector method), 1223

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchVector.NumberedChromaticPitchVector.NumberedChromaticPitchVector method), 1246

__getitem__() (abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector method), 1116

__getitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1253

__getitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1259

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1265

__getitem__() (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer method), 1336

__getitem__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1376

__getitem__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1384

__getitem__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1393

__getitem__() (abjad.tools.scoretools.Score.Score.Score method), 1402

__getitem__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1411

__getitem__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1419

__getitem__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1662

__getitem__() (abjad.tools.stafftools.Staff.Staff.Staff method), 1671

__getitem__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1710

__getitem__() (abjad.tools.timeintervaltools.TimeIntervalTree.Dictionary(Dictionary(Dictionary method), 1744

__getitem__() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1839

__getitem__() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1848

__getitem__() (abjad.tools.voicetools.Voice.Voice.Voice method), 1873

__setslice__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 383

__setslice__() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 417

__setitem__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1907

__setitem__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 661

__setitem__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector method), 865

__setitem__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector method), 868

__setitem__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 878

__setitem__() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector method), 861

__setitem__() (abjad.tools.pitchtools.NumberedChromaticPitchVector.NumberedChromaticPitchVector.NumberedChromaticPitchVector method), 886

__setitem__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 939

__setitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1254

__setitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1259

__setitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1265

__setitem__() (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer method), 1385

__setitem__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1420

__setitem__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1975

__setitem__() (abjad.tools.exceptiontools.ClefError.ClefError.ClefError method), 1975

__setstate__() (abjad.tools.exceptiontools.ContainmentError.ContainmentError.ContainmentError method), 1976

__setstate__() (abjad.tools.exceptiontools.ContextContainmentError.ContextContainmentError.ContextContainmentError method), 1977

__setstate__() (abjad.tools.exceptiontools.ContiguityError.ContiguityError.ContiguityError method), 1978

__setstate__() (abjad.tools.exceptiontools.CyclicNodeError.CyclicNodeError.CyclicNodeError method), 1979

__setstate__() (abjad.tools.exceptiontools.DurationError.DurationError.DurationError method), 1980

__setstate__() (abjad.tools.exceptiontools.ExtraMarkError.ExtraMarkError.ExtraMarkError method), 1981

__setstate__() (abjad.tools.exceptiontools.ExtraNoteHeadError.ExtraNoteHeadError.ExtraNoteHeadError method), 1982

__setstate__() (abjad.tools.exceptiontools.ExtraPitchError.ExtraPitchError.ExtraPitchError method), 1983

__setstate__() (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.ExtraSpannerError method), 1984

__setstate__() (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.ExtraSpannerError method), 1985

__setstate__() (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.ExtraSpannerError method), 1986

__setstate__() (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError.ExtraSpannerError method), 1987

[__setstate__\(\)](#) (abjad.tools.exceptiontools.InputSpecificationError.InputSpecificationError.exceptiontools.SpacingError.SpacingError method), 1988
[__setstate__\(\)](#) (abjad.tools.exceptiontools.InstrumentError.InstrumentError.exceptiontools.SpannerError.SpannerError method), 1989
[__setstate__\(\)](#) (abjad.tools.exceptiontools.IntervalError.IntervalError.exceptiontools.SpannerPopulationError.SpannerPopulationError method), 1990
[__setstate__\(\)](#) (abjad.tools.exceptiontools.LilyPondParserError.LilyPondParserError.exceptiontools.StaffContainmentError.StaffContainmentError method), 1991
[__setstate__\(\)](#) (abjad.tools.exceptiontools.LineBreakError.LineBreakError.exceptiontools.TempoError.TempoError method), 1992
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MarkError.MarkError.exceptiontools.TieChainError.TieChainError method), 1993
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MeasureContiguityError.MeasureContiguityError.exceptiontools.TimeSignatureAssignmentError.TimeSignatureAssignmentError method), 1994
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MeasureError.MeasureError.exceptiontools.TimeSignatureError.TimeSignatureError method), 1995
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MissingComponentError.MissingComponentError.exceptiontools.TonalHarmonyError.TonalHarmonyError method), 1996
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MissingInstrumentError.MissingInstrumentError.exceptiontools.TupletError.TupletError method), 1997
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MissingMarkError.MissingMarkError.exceptiontools.TupletFuseError.TupletFuseError method), 1998
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MissingMeasureError.MissingMeasureError.exceptiontools.TypographicWhitespaceError.TypographicWhitespaceError method), 1999
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MissingNamedComponentError.MissingNamedComponentError.exceptiontools.UnboundedTimeIntervalError.UnboundedTimeIntervalError method), 2000
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MissingNoteHeadError.MissingNoteHeadError.exceptiontools.UndefinedSpacingError.UndefinedSpacingError method), 2001
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MissingPitchError.MissingPitchError.exceptiontools.UnderfullContainerError.UnderfullContainerError method), 2002
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MissingSpannerError.MissingSpannerError.exceptiontools.VoiceContainmentError.VoiceContainmentError method), 2003
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MissingTempoError.MissingTempoError.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer method), 2004
[__setstate__\(\)](#) (abjad.tools.exceptiontools.MusicContentsError.MusicContentsError.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf method), 2005
[__setstate__\(\)](#) (abjad.tools.exceptiontools.NegativeDurationError.NegativeDurationError.rhythmtreetools.RhythmTreeNode.RhythmTreeNode method), 2006
[__setstate__\(\)](#) (abjad.tools.exceptiontools.NonbinaryTimeSignatureConvergenceError.NonbinaryTimeSignatureConvergenceError method), 2007
[__setstate__\(\)](#) (abjad.tools.exceptiontools.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError method), 2008
[__setstate__\(\)](#) (abjad.tools.exceptiontools.NoteHeadError.NoteHeadError.abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.K method), 2009
[__setstate__\(\)](#) (abjad.tools.exceptiontools.OverfullContainerError.OverfullContainerError.abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark method), 2010
[__setstate__\(\)](#) (abjad.tools.exceptiontools.ParallelError.ParallelError(abjad.tools.durationtools.Duration.Duration.Duration method), 2011
[__setstate__\(\)](#) (abjad.tools.exceptiontools.PartitionError.PartitionError(abjad.tools.durationtools.Offset.Offset.Offset method), 2012
[__setstate__\(\)](#) (abjad.tools.exceptiontools.PitchError.PitchError.__setstate__()(abjad.tools.exceptiontools.AssignabilityError.AssignabilityError method), 2013
[__setstate__\(\)](#) (abjad.tools.exceptiontools.SchemeParserFinishedException.SchemeParserFinishedException.abjad.tools.clefError.ClefError method), 2014

__str__() (abjad.tools.exceptiontools.ContainmentError.ContainmentError method), 1976
 __str__() (abjad.tools.exceptiontools.ContextContainmentError.ContextContainmentError method), 1977
 __str__() (abjad.tools.exceptiontools.ContiguityError.ContiguityError method), 1978
 __str__() (abjad.tools.exceptiontools.CyclicNodeError.CyclicNodeError method), 1979
 __str__() (abjad.tools.exceptiontools.DurationError.DurationError method), 1980
 __str__() (abjad.tools.exceptiontools.ExtraMarkError.ExtraMarkError method), 1981
 __str__() (abjad.tools.exceptiontools.ExtraNamedComponentError.ExtraNamedComponentError method), 1982
 __str__() (abjad.tools.exceptiontools.ExtraNoteHeadError.ExtraNoteHeadError method), 1983
 __str__() (abjad.tools.exceptiontools.ExtraPitchError.ExtraPitchError method), 1984
 __str__() (abjad.tools.exceptiontools.ExtraSpannerError.ExtraSpannerError method), 1985
 __str__() (abjad.tools.exceptiontools.GraceContainerError.GraceContainerError method), 1986
 __str__() (abjad.tools.exceptiontools.ImpreciseTempoError.ImpreciseTempoError method), 1987
 __str__() (abjad.tools.exceptiontools.InputSpecificationError.InputSpecificationError method), 1988
 __str__() (abjad.tools.exceptiontools.InstrumentError.InstrumentError method), 1989
 __str__() (abjad.tools.exceptiontools.IntervalError.IntervalError method), 1990
 __str__() (abjad.tools.exceptiontools.LilyPondParserError.LilyPondParserError method), 1991
 __str__() (abjad.tools.exceptiontools.LineBreakError.LineBreakError method), 1992
 __str__() (abjad.tools.exceptiontools.MarkError.MarkError method), 1993
 __str__() (abjad.tools.exceptiontools.MeasureContiguityError.MeasureContiguityError method), 1994
 __str__() (abjad.tools.exceptiontools.MeasureError.MeasureError method), 1995
 __str__() (abjad.tools.exceptiontools.MissingComponentError.MissingComponentError method), 1996
 __str__() (abjad.tools.exceptiontools.MissingInstrumentError.MissingInstrumentError method), 1997
 __str__() (abjad.tools.exceptiontools.MissingMarkError.MissingMarkError method), 1998
 __str__() (abjad.tools.exceptiontools.MissingMeasureError.MissingMeasureError method), 1999
 __str__() (abjad.tools.exceptiontools.MissingNamedComponentError.MissingNamedComponentError method), 2000
 __str__() (abjad.tools.exceptiontools.MissingNoteHeadError.MissingNoteHeadError method), 2001
 __str__() (abjad.tools.exceptiontools.MissingPitchError.MissingPitchError method), 2002
 __str__() (abjad.tools.exceptiontools.MissingSpannerError.MissingSpannerError method), 2003
 __str__() (abjad.tools.exceptiontools.MissingTempoError.MissingTempoError method), 2004
 __str__() (abjad.tools.exceptiontools.MusicContentsError.MusicContentsError method), 2005
 __str__() (abjad.tools.exceptiontools.NegativeDurationError.NegativeDurationError method), 2006
 __str__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError method), 2007
 __str__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError method), 2008
 __str__() (abjad.tools.exceptiontools.NoteHeadError.NoteHeadError method), 2009
 __str__() (abjad.tools.exceptiontools.OverfullContainerError.OverfullContainerError method), 2010
 __str__() (abjad.tools.exceptiontools.ParallelError.ParallelError method), 2011
 __str__() (abjad.tools.exceptiontools.PartitionError.PartitionError method), 2012
 __str__() (abjad.tools.exceptiontools.PitchError.PitchError method), 2013
 __str__() (abjad.tools.exceptiontools.SchemeParserFinishedException.SchemeParserFinishedException method), 2014
 __str__() (abjad.tools.exceptiontools.SpacingError.SpacingError method), 2015
 __str__() (abjad.tools.exceptiontools.SpannerError.SpannerError method), 2016
 __str__() (abjad.tools.exceptiontools.SpannerPopulationError.SpannerPopulationError method), 2017
 __str__() (abjad.tools.exceptiontools.StaffContainmentError.StaffContainmentError method), 2018
 __str__() (abjad.tools.exceptiontools.TempoError.TempoError method), 2019
 __str__() (abjad.tools.exceptiontools.TieChainError.TieChainError method), 2020
 __str__() (abjad.tools.exceptiontools.TimeSignatureAssignmentError.TimeSignatureAssignmentError method), 2021
 __str__() (abjad.tools.exceptiontools.TimeSignatureError.TimeSignatureError method), 2022
 __str__() (abjad.tools.exceptiontools.TonalHarmonyError.TonalHarmonyError method), 2023
 __str__() (abjad.tools.exceptiontools.TupletError.TupletError method), 2024
 __str__() (abjad.tools.exceptiontools.TupletFuseError.TupletFuseError method), 2025
 __str__() (abjad.tools.exceptiontools.TypographicWhitespaceError.TypographicWhitespaceError method), 2026
 __str__() (abjad.tools.exceptiontools.UnboundedTimeIntervalError.UnboundedTimeIntervalError method), 2027
 __str__() (abjad.tools.exceptiontools.UndefinedSpacingError.UndefinedSpacingError method), 2028
 __str__() (abjad.tools.exceptiontools.UnderfullContainerError.UnderfullContainerError method), 2029

__str__() (abjad.tools.exceptiontools.VoiceContainmentError.VoiceContainmentError method), 2030

__str__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 830

__str__() (abjad.tools.lilypondparsertools.SyntaxNode.SyntaxNode.SyntaxNode method), 2079

__str__() (abjad.tools.marktools.Articulation.Articulation.Articulation method), 895

__str__() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 902

__str__() (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method), 913

__str__() (abjad.tools.markuptools.Markup.Markup.Markup method), 934

__str__() (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand method), 936

__str__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction method), 952

__str__() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 988

__str__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 999

__str__() (abjad.tools.measuretools.Measure.Measure.Measure method), 1009

__str__() (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 1036

__str__() (abjad.tools.notetools.Note.Note.Note method), 1040

__str__() (abjad.tools.notetools.NoteHead.NoteHead.NoteHead method), 1043

__str__() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method), 1051

__str__() (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell method), 1054

__str__() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1056

__str__() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1058

__str__() (abjad.tools.pitchtools.Accidental.Accidental.Accidental method), 1130

__str__() (abjad.tools.pitchtools.ChromaticIntervalClassObjects.ChromaticIntervalClassObjects.ChromaticIntervalClassObjects method), 1065

__str__() (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.ChromaticIntervalObject method), 1067

__str__() (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject.CounterpointIntervalClassObject method), 1071

__str__() (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.CounterpointIntervalObject method), 1073

__str__() (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject.DiatonicIntervalClassObject method), 1076

__str__() (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.DiatonicIntervalObject method), 1077

__str__() (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval method), 1132

__str__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 1133

__str__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1137

__str__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1139

__str__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1142

__str__() (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 1143

__str__() (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1145

__str__() (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval method), 1146

__str__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1148

__str__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1151

__str__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1154

__str__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1157

__str__() (abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject.HarmonicIntervalClassObject method), 1083

__str__() (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.HarmonicIntervalObject method), 1085

__str__() (abjad.tools.pitchtools.IntervalObject.IntervalObject.IntervalObject method), 1093

__str__() (abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass.IntervalObjectClass method), 1094

__str__() (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment.IntervalObjectSegment method), 1096

__str__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1158

__str__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass method), 1168

__str__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet method), 1171

__str__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector method), 1174

__str__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassObject.MelodicChromaticIntervalClassObject.MelodicChromaticIntervalClassObject method), 1176

__str__() (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1177

__str__() (abjad.tools.pitchtools.MelodicCounterpointIntervalClassObject.MelodicCounterpointIntervalClassObject.MelodicCounterpointIntervalClassObject method), 1182

__str__() (abjad.tools.pitchtools.MelodicCounterpointIntervalClassVector.MelodicCounterpointIntervalClassVector.MelodicCounterpointIntervalClassVector method), 1185

__str__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSet.MelodicChromaticIntervalClassSet.MelodicChromaticIntervalClassSet method), 1188

__str__() (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval method), 1190

__str__() (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1192

__str__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.SchemePair.SchemePair method), 1195

__str__() (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.SchemeVector.SchemeVector method), 1196

__str__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 1199

__str__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.CyclicList method), 1202

__str__() (abjad.tools.pitchtools.MelodicIntervalClassObject.MelodicIntervalClassObject.MelodicIntervalClassObject.CyclicTree method), 1101

__str__() (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject.MelodicIntervalObject.CyclicTuple.CyclicTuple method), 1103

__str__() (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch.Tree.Tree.Tree method), 1206

__str__() (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1209

__str__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1211

__str__() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1214

__str__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet.ChordQualityIndicator.ChordQualityIndicator method), 1220

__str__() (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch.Mode.Mode.Mode method), 1227

__str__() (abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass.NamedDiatonicPitchClass.Scale method), 1229

__str__() (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch.NumberedChromaticPitch.ScaleDegree.ScaleDegree method), 1232

__str__() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1234

__str__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1238

__str__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1243

__str__() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector method), 1246

__str__() (abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch.NumberedDiatonicPitch.TempoMark.TempoMark.TempoMark method), 1248

__str__() (abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass method), 1250

__str__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 1268

__str__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest method), 1310

__str__() (abjad.tools.resttools.Rest.Rest.Rest method), 1313

__str__() (abjad.tools.schemetools.Scheme.Scheme.Scheme method), 1348

__str__() (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList method), 1350

__str__() (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor method), 1352

__str__() (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment method), 1353

__str__() (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction method), 953

__str__() (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 1036

__str__() (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList method), 1040

__str__() (abjad.tools.pitchtools.Accidental.Accidental.Accidental method), 1130

__str__() (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.ChromaticIntervalObject method), 1067

__sub__() (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval method), 1132

__sub__() (abjad.tools.pitchtools.HarmonicChromaticInterval.IntervalClassObjectSet.IntervalClassObjectSet method), 1142

__sub__() (abjad.tools.pitchtools.HarmonicDiatonicInterval.IntervalClassObjectSet.IntervalClassObjectSet method), 1151

__sub__() (abjad.tools.pitchtools.HarmonicDiatonicInterval.IntervalClassObjectSet.IntervalClassObjectSet method), 1157

__sub__() (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet method), 1091

__sub__() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet method), 1099

__sub__() (abjad.tools.pitchtools.InversionEquivalentChromaticInterval.IntervalClassObjectSet.IntervalClassObjectSet method), 1163

__sub__() (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval method), 1176

__sub__() (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval method), 1188

__sub__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1195

__sub__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1202

__sub__() (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch method), 1206

__sub__() (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1209

__sub__() (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1214

__sub__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1220

__sub__() (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch method), 1232

__sub__() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1234

__sub__() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1243

__sub__() (abjad.tools.pitchtools.ObjectSet.ObjectSet method), 1113

__sub__() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet method), 1122

__sub__() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet method), 1128

__sub__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest method), 1310

__sub__() (abjad.tools.resttools.Rest.Rest.Rest method), 1313

__sub__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1497

__sub__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1806

__truediv__() (abjad.tools.durationtools.Duration.Duration.Duration method), 448

__truediv__() (abjad.tools.durationtools.Offset.Offset.Offset method), 452

__unicode__() (abjad.tools.exceptiontools.InputSpecificationError.InputSpecificationError method), 1888

__unicode__() (abjad.tools.exceptiontools.InstrumentError.InstrumentError method), 1889

__unicode__() (abjad.tools.exceptiontools.IntervalError.IntervalError method), 1990

__unicode__() (abjad.tools.exceptiontools.LilyPondParserError.LilyPondParserError method), 1991

__unicode__() (abjad.tools.exceptiontools.LineBreakError.LineBreakError method), 1992

__unicode__() (abjad.tools.exceptiontools.MarkError.MarkError method), 1993

__unicode__() (abjad.tools.exceptiontools.MeasureContiguityError.MeasureContiguityError method), 1994

__unicode__() (abjad.tools.exceptiontools.MeasureError.MeasureError method), 1995

__unicode__() (abjad.tools.exceptiontools.MissingComponentError.MissingComponentError method), 1996

__unicode__() (abjad.tools.exceptiontools.MissingInstrumentError.MissingInstrumentError.exceptiontools.TupletError.TupletError method), 1997

__unicode__() (abjad.tools.exceptiontools.MissingMarkError.MissingMarkError.abjad.tools.exceptiontools.TupletFuseError.TupletFuseError method), 1998

__unicode__() (abjad.tools.exceptiontools.MissingMeasureError.MissingMeasureError.abjad.tools.exceptiontools.TypographicWhitespaceError.TypographicWhitespaceError method), 1999

__unicode__() (abjad.tools.exceptiontools.MissingNamedComponentError.MissingNamedComponentError.abjad.tools.exceptiontools.UnboundedTimeIntervalError.UnboundedTimeIntervalError method), 2000

__unicode__() (abjad.tools.exceptiontools.MissingNoteHeadError.MissingNoteHeadError.abjad.tools.exceptiontools.UndefinedSpacingError.UndefinedSpacingError method), 2001

__unicode__() (abjad.tools.exceptiontools.MissingPitchError.MissingPitchError.abjad.tools.exceptiontools.UnderfullContainerError.UnderfullContainerError method), 2002

__unicode__() (abjad.tools.exceptiontools.MissingSpannerError.MissingSpannerError.abjad.tools.exceptiontools.VoiceContainmentError.VoiceContainmentError method), 2003

__unicode__() (abjad.tools.exceptiontools.MissingTempoError.MissingTempoError.abjad.tools.chordtools.Chord.Chord.Chord method), 2004

__unicode__() (abjad.tools.exceptiontools.MusicContentsError.MusicContentsError.abjad.tools.leaftools.Leaf.Leaf.Leaf method), 2005

__unicode__() (abjad.tools.exceptiontools.NegativeDurationError.NegativeDurationError.abjad.tools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 2006

__unicode__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureConversionError.NonbinaryTimeSignatureConversionError.abjad.tools.NonbinaryTimeSignatureConversionError method), 2007

__unicode__() (abjad.tools.exceptiontools.NonbinaryTimeSignatureSuppressionError.NonbinaryTimeSignatureSuppressionError.abjad.tools.NonbinaryTimeSignatureSuppressionError method), 2008

__unicode__() (abjad.tools.exceptiontools.NoteHeadError.NoteHeadError.abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 2009

__unicode__() (abjad.tools.exceptiontools.OverfullContainerError.OverfullContainerError.abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 2010

__unicode__() (abjad.tools.exceptiontools.ParallelError.ParallelError(abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet method), 2011

__unicode__() (abjad.tools.exceptiontools.PartitionError.PartitionError(abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet method), 2012

__unicode__() (abjad.tools.exceptiontools.PitchError.PitchError(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 2013

__unicode__() (abjad.tools.exceptiontools.SchemeParserFinishedException(abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 2014

__unicode__() (abjad.tools.exceptiontools.SpacingError.SpacingError(abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 2015

__unicode__() (abjad.tools.exceptiontools.SpannerError.SpannerError(abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 2016

__unicode__() (abjad.tools.exceptiontools.SpannerPopulationError.SpannerPopulationError(abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet method), 2017

__unicode__() (abjad.tools.exceptiontools.StaffContainmentError.StaffContainmentError(abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 2018

__unicode__() (abjad.tools.exceptiontools.TempoError.TempoError(abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet method), 2019

__unicode__() (abjad.tools.exceptiontools.TieChainError.TieChainError(abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 2020

__unicode__() (abjad.tools.exceptiontools.TimeSignatureAssignmentError(abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet method), 2021

__unicode__() (abjad.tools.exceptiontools.TimeSignatureError.TimeSignatureError(abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest method), 2022

__unicode__() (abjad.tools.exceptiontools.TonalHarmonyError.TonalHarmonyError(abjad.tools.resttools.Rest.Rest.Rest method), 2023

__xor__() (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass.HTMLOutputFormat (class in abjad.tools.abjadbooktools.HTMLOutputFormat.HTMLOutputFormat), 1492

__xor__() (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression.HTMLOutputFormat (class in abjad.tools.abjadbooktools.HTMLOutputFormat.HTMLOutputFormat), 1494

__xor__() (abjad.tools.skiptools.Skip.Skip.Skip method), abjad.tools.abjadbooktools.LaTeXOutputFormat.LaTeXOutputFormat, 1497 1891

__xor__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.OutputFormat (class in abjad.tools.abjadbooktools.OutputFormat.OutputFormat), 1806 1883

A

abctools.AbjadObject (class in abjad.tools.abctools.AbjadObject.AbjadObject), abjadbooktools.ReSTOutputFormat (class in abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutputFormat), 1875 1893

abctools.AttributeEqualityAbjadObject (class in abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject), accepts (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock attribute), 1876 869

abctools.ImmutableAbjadObject (class in abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject), accidental (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree attribute), 1877 1819

abctools.Parser (class in abjad.tools.abctools.Parser.Parser), accidental_spelling (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch attribute), 1879 1203

abctools.ScoreSelection (class in abjad.tools.abctools.ScoreSelection.ScoreSelection), alias (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.AbjadBookScript attribute), 1881 1887

abctools.SortableAttributeEqualityAbjadObject alias (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript attribute), 1882 1915

ABJAD_CONFIG_DIRECTORY_PATH (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig attribute), 1895 1918

ABJAD_CONFIG_FILE_PATH (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig attribute), 1895 1922

ABJAD_EXPERIMENTAL_PATH (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig attribute), 1895 1927

ABJAD_OUTPUT_PATH (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig attribute), 1895 1912

ABJAD_PATH (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig attribute), 1896 1938

ABJAD_ROOT_PATH (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig attribute), 1896 1934

abjadbooktools.AbjadBookProcessor (class in abjad.tools.abjadbooktools.AbjadBookProcessor.AbjadBookProcessor), alias (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript attribute), 1885 1937

abjadbooktools.AbjadBookScript (class in abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript), alias (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript attribute), 1886 1940

abjadbooktools.CodeBlock (class in abjad.tools.abjadbooktools.CodeBlock.CodeBlock), alias (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript attribute), 1888 1944

alias (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript attribute), 1944

alias (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript attribute), 1944

attribute), 1129

always_format_time_signature (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.Measure.Measure.Measure attribute), 982

always_format_time_signature (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 993

always_format_time_signature (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1004

append() (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner method), 225

append() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 233

append() (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 242

append() (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 250

append() (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 257

append() (abjad.tools.chordtools.Chord.Chord.Chord method), 269

append() (abjad.tools.containertools.Cluster.Cluster.Cluster method), 338

append() (abjad.tools.containertools.Container.Container.Container method), 344

append() (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer method), 352

append() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 381

append() (abjad.tools.contexttools.Context.Context.Context method), 388

append() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 415

append() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1905

append() (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 478

append() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 659

append() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 863

append() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 866

append() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 875

append() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 859

append() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 884

append() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 937

append() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure.Measure.Measure.Measure attribute), 984

append() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 995

append() (abjad.tools.measuretools.Measure.Measure.Measure method), 1005

append() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1057

append() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1057

append() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1251

append() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1257

append() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1263

append() (abjad.tools.pitchtools.RhythmSpanner.RhythmSpanner.RhythmSpanner method), 1329

append() (abjad.tools.pitchtools.StaffGroup.StaffGroup.StaffGroup method), 1372

append() (abjad.tools.piano.PianoStaff.PianoStaff.PianoStaff method), 1382

append() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1389

append() (abjad.tools.scoretools.Score.Score.Score method), 1398

append() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1407

append() (abjad.tools.scoretools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer method), 1418

append() (abjad.tools.scoretools.BracketSpanner.BracketSpanner.BracketSpanner method), 1514

append() (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner method), 1521

append() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1530

append() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1538

append() (abjad.tools.spannertools.DirectedSpanner.DirectedSpanner.DirectedSpanner method), 1502

append() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1545

append() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1551

append() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1559

append() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1566

append() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1572

append() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method), 1579

append() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner method), 1586

append() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1593

attribute), 1918

argument_parser(abjad.tools.developerscripttools.BuildApiScript), 496

attribute), 1920

argument_parser(abjad.tools.developerscripttools.CleanScript), 502

attribute), 1923

argument_parser(abjad.tools.developerscripttools.CountLines), 508

attribute), 1925

argument_parser(abjad.tools.developerscripttools.CountTools), 520

attribute), 1927

argument_parser(abjad.tools.developerscripttools.DevelopScript), 525

attribute), 1910

argument_parser(abjad.tools.developerscripttools.DirectoryScript), 532

attribute), 1912

argument_parser(abjad.tools.developerscripttools.MakeNewClass), 538

attribute), 1930

argument_parser(abjad.tools.developerscripttools.MakeNewClass), 562

attribute), 1932

argument_parser(abjad.tools.developerscripttools.RenameModule), 544

attribute), 1934

argument_parser(abjad.tools.developerscripttools.ReplaceIncludes), 550

attribute), 1937

argument_parser(abjad.tools.developerscripttools.ReplacePatterns), 555

attribute), 1940

argument_parser(abjad.tools.developerscripttools.RunDoctestScript), 514

attribute), 1942

argument_parser(abjad.tools.developerscripttools.SvnAddAllScript), 514

attribute), 1944

argument_parser(abjad.tools.developerscripttools.SvnCommitScript), 567

attribute), 1947

argument_parser(abjad.tools.developerscripttools.SvnMessageScript), 574

attribute), 1949

argument_parser(abjad.tools.developerscripttools.SvnUpdateScript), 579

arguments(abjad.tools.documentationtools.Pipe.Pipe.Pipe), 586

attribute), 1970

attach()(abjad.tools.contexttools.ClefMark.ClefMark.ClefMark), 592

method), 379

attach()(abjad.tools.contexttools.ContextMark.ContextMark.ContextMark), 604

method), 393

attach()(abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark), 609

method), 397

attach()(abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark), 616

method), 401

attach()(abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark), 622

method), 404

attach()(abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark), 627

method), 408

attach()(abjad.tools.contexttools.TempoMark.TempoMark.TempoMark), 633

method), 413

attach()(abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark), 638

method), 421

attach()(abjad.tools.instrumenttools.Accordion.Accordion.Accordion), 644

method), 489

attach()(abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute), 651

attach()(abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone), 651

attach()(abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone), 651

attach()(abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet), 651

attach()(abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute), 651

attach()(abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone), 651

attach()(abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone), 651

attach()(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice), 651

attach()(abjad.tools.instrumenttools.BellClarinets.BellClarinets.BellClarinets), 651

attach()(abjad.tools.instrumenttools.Cello.Cello.Cello), 651

attach()(abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass), 651

attach()(abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute), 651

attach()(abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon), 651

attach()(abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone), 651

attach()(abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice), 651

attach()(abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet), 651

attach()(abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn), 651

attach()(abjad.tools.instrumenttools.Flute.Flute.Flute), 651

attach()(abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn), 651

attach()(abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel), 651

attach()(abjad.tools.instrumenttools.Guitar.Guitar.Guitar), 651

attach()(abjad.tools.instrumenttools.Harp.Harp.Harp), 651

attach()(abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord), 651

2203

check() (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck, method), 2094

check() (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.MissingParentCheck), 2096

check() (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck), 2097

check() (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck), 2099

check() (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck), 2100

check() (abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.OverlappingOctavationCheck), 2102

check() (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck), 2103

child_graph (abjad.tools.datastructuretools.Digraph.Digraph.Digraph attribute), 1224

children (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer attribute), 1324

children (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1424

children (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1437

chord_name (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator attribute), 1820

chordtools.all_are_chords() (in module abjad.tools.chordtools.all_are_chords), 271

chordtools.arpeggiate_chord() (in module abjad.tools.chordtools.arpeggiate_chord), 272

chordtools.change_defective_chord_to_note_or_rest() (in module abjad.tools.chordtools.change_defective_chord_to_note_or_rest), 272

chordtools.Chord (class in abjad.tools.chordtools.Chord.Chord), 266

chordtools.divide_chord_by_chromatic_pitch_number() (in module abjad.tools.chordtools.divide_chord_by_chromatic_pitch_number), 273

chordtools.divide_chord_by_diatonic_pitch_number() (in module abjad.tools.chordtools.divide_chord_by_diatonic_pitch_number), 273

chordtools.get_arithmetic_mean_of_chord() (in module abjad.tools.chordtools.get_arithmetic_mean_of_chord), 274

chordtools.get_note_head_from_chord_by_pitch() (in module abjad.tools.chordtools.get_note_head_from_chord_by_pitch), 274

chordtools.make_tied_chord() (in module abjad.tools.chordtools.make_tied_chord), 274

chordtools.yield_all_subchords_of_chord() (in module abjad.tools.chordtools.yield_all_subchords_of_chord), 274

chromatic_pitch_name (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch attribute), 1203

chromatic_pitch_class_name (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch attribute), 1203

chromatic_pitch_class_number (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch attribute), 1203

chromatic_pitch_class_numbers (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch attribute), 1203

chromatic_pitch_number (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch attribute), 1203

chromatic_pitch_numbers (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1215

chromatic_pitch_set (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet attribute), 1218

chromatic_pitch_vector (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector attribute), 1221

clear() (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner method), 226

clear() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner method), 226

	(method),	233
clear()	(abjad.tools.beamttools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.Pia	
	(method),	242
clear()	(abjad.tools.beamttools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.	
	(method),	251
clear()	(abjad.tools.beamttools.MultipartBeamSpanner.MultipartBeamSpanner.Spanner.Spanner	
	(method),	257
clear()	(abjad.tools.chordtools.Chord.Chord.Chord clear() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.Staff	
	(method),	269
clear()	(abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig) (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextS	
	(method),	1896
clear()	(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary) (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner	
	(method),	1903
clear()	(abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph) (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner	
	(method),	1966
clear()	(abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector) (abjad.tools.pitchtools.IntervalClassSpanner.IntervalClassSpanner.HarmonicChromaticInter	
	(method),	1135
clear()	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector) (abjad.tools.pitchtools.IntervalClassSpanner.IntervalClassSpanner.NeotonInterva	
	(method),	1165
clear()	(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector) (abjad.tools.pitchtools.IntervalClassSpanner.IntervalClassSpanner.VaryOfInversio	
	(method),	1172
clear()	(abjad.tools.pitchtools.MelodicChromaticIntervalClassVector) (abjad.tools.pitchtools.IntervalClassSpanner.IntervalClassSpanner.GuidedPrevailC	
	(method),	1181
clear()	(abjad.tools.pitchtools.NamedChromaticPitch Vector.NamedChromaticPitch Vector.NeotonMark.ClefMark.ClefMark.GrefMark	
	(method),	1222
clear()	(abjad.tools.pitchtools.NumberedChromaticPitch Class.Vector.NumberedChromaticPitch Class.Vector.Pipe.Pipe.Pipe ChromaticPitchCl	
	(method),	1244
clear()	(abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector code_block_closing (ab-	
	(method),	1114
clear()	(abjad.tools.spannertools.BraceSpanner.BraceSpanner.BraceSpanner code_block_closing (ab-	
	(method),	1514
clear()	(abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.LaTeXOutputForm	
	(method),	1521
clear()	(abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner code_block_closing (ab-	
	(method),	1530
clear()	(abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner code_block_closing (ab-	
	(method),	1538
clear()	(abjad.tools.spannertoolsDirectedSpannerDirectedSpannerDirectedSpanner directed_spanner_book_tools.ReSTOutputFormat.ReSTOutputFormat	
	(method),	1502
clear()	(abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpannerDynamicTextSpanner (ab-	
	(method),	1545
clear()	(abjad.tools.spannertoolsGlissandoSpannerGlissandoSpannerGlissandoSpanner (ab-	
	(method),	1552
clear()	(abjad.tools.spannertoolsHairpinSpannerHairpinSpannerHairpinSpanner hairpin_spacer_abjadbooktools.LaTeXOutputFormat.LaTeXOutputForm	
	(method),	1560
clear()	(abjad.tools.spannertools.HiddenStaffSpannerHiddenStaffSpannerHiddenStaffSpanner (ab-	
	(method),	1566
clear()	(abjad.tools.spannertools.HorizontalBracketSpannerHorizontalBracketSpannerHorizontalBracketSpanner	
	(method),	1572
clear()	(abjad.tools.spannertools.MetricGridSpannerMetricGridSpannerMetricGridSpannerReSTOutputFormat.ReSTOutputFormat	
	(method),	1579
clear()	(abjad.tools.spannertools.OctavationSpannerOctavationSpannerOctavationSpanner abjadbooktools.HTMLOutputFormat.HTMLOutpu	
	(method),	1587
clear()	(abjad.tools.spannertoolsPhrasingSlurSpannerPhrasingSlurSpannerPhrasingSlurSpannerLaTeXOutputFormat.LaTeXOutpu	

attribute), 1892
 code_indent (abjad.tools.abjadbooktools.OutputFormat.OutputFormat attribute), 1884
 code_indent (abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutputFormat attribute), 1893
 code_root (abjad.tools.documentationtools.APICrawler.APICrawler attribute), 1954
 code_root (abjad.tools.documentationtools.ClassCrawler.ClassCrawler attribute), 1957
 code_root (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler attribute), 1962
 code_root (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler attribute), 1968
 code_tools_path (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator attribute), 1955
 colors (abjad.tools.pitchtools.NumberedChromaticPitchClassColorMap.NumberedChromaticPitchClassColorMap attribute), 1234
 column_index (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn attribute), 1055
 column_indices (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell attribute), 1053
 columns (abjad.tools.pitcharraytools.PitchArray.PitchArray attribute), 1050
 columns (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix attribute), 1421
 columns (abjad.tools.sequencetools.Matrix.Matrix attribute), 1434
 command (abjad.tools.markuptools.MarkupCommand.MarkupCommand attribute), 935
 command_name (abjad.tools.marktools.BarLine.BarLine attribute), 897
 command_name (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark attribute), 904
 commit_message_path (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript attribute), 1949
 communicate() (abjad.tools.documentationtools.Pipe.Pipe method), 1970
 components (abjad.tools.beamtools.BeamSpanner.BeamSpanner attribute), 224
 components (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner attribute), 231
 components (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 239
 components (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 248
 components (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner attribute), 256
 components (abjad.tools.spannertools.BraceSpanner.BraceSpanner attribute), 1513
 components (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner attribute), 1519
 components (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner attribute), 1527
 components (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner attribute), 1536
 components (abjad.tools.spannertoolsDirectedSpannerDirectedSpanner attribute), 1537
 components (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner attribute), 1543
 components (abjad.tools.spannertoolsGlissandoSpannerGlissandoSpanner attribute), 1550
 components (abjad.tools.spannertoolsHairpinSpannerHairpinSpanner attribute), 1557
 components (abjad.tools.spannertoolsHiddenStaffSpannerHiddenStaffSpanner attribute), 1569
 components (abjad.tools.spannertoolsHorizontalBracketSpannerHorizontalBracketSpanner attribute), 1571
 components (abjad.tools.spannertoolsMetricGridSpannerMetricGridSpanner attribute), 1577
 components (abjad.tools.spannertoolsOctavationSpannerOctavationSpanner attribute), 1583
 components (abjad.tools.spannertoolsPhrasingSlurSpannerPhrasingSlurSpanner attribute), 1598
 components (abjad.tools.spannertoolsPianoPedalSpannerPianoPedalSpanner attribute), 1605
 components (abjad.tools.spannertoolsSlurSpannerSlurSpanner attribute), 1605
 components (abjad.tools.spannertoolsSpannerSpanner attribute), 1506
 components (abjad.tools.spannertoolsStaffLinesSpannerStaffLinesSpanner attribute), 1618
 components (abjad.tools.spannertoolsTextScriptSpannerTextScriptSpanner attribute), 1622
 components (abjad.tools.spannertoolsTextSpannerTextSpanner attribute), 1631
 components (abjad.tools.spannertoolsTieSpannerTieSpanner attribute), 1685
 components (abjad.tools.spannertoolsTrillSpannerTrillSpanner attribute), 1859
 components (abjad.tools.spannertoolsVerticalMomentSpannerVerticalMomentSpanner attribute), 1859
 components_are_components() (in module abjad.tools.componenttools.all_are_components), 239
 components_are_components_in_same_parent() (in module abjad.tools.componenttools.all_are_components_in_same_parent), 239
 components_are_components_in_same_score() (in module abjad.tools.componenttools.all_are_components_in_same_score), 239
 components_are_components_in_same_thread() (in module abjad.tools.componenttools.all_are_components_in_same_thread), 239
 components_scalable_by_multiplier() (in module abjad.tools.componenttools.all_are_components_scalable_by_multiplier), 239

(in module ab- componenttools.component_to_tuplet_depth()
 jad.tools.componenttools.all_are_components_scalable_by_(multiplier), module ab-
 280 jad.tools.componenttools.component_to_tuplet_depth()),
 componenttools.all_are_contiguous_components() 286
 (in module ab- componenttools.ContainmentSignature (class in ab-
 jad.tools.componenttools.all_are_contiguous_components), jad.tools.componenttools.ContainmentSignature.ContainmentSig
 281 278
 componenttools.all_are_contiguous_components_in_same_parent() componenttools.copy_and_partition_governed_component_subtree_by_leaf
 (in module ab- (in module ab-
 jad.tools.componenttools.all_are_contiguous_components_in_same_parent() jad.tools.componenttools.copy_and_partition_governed_componen
 281 286
 componenttools.all_are_contiguous_components_in_same_score() componenttools.copy_components_and_covered_spanners()
 (in module ab- (in module ab-
 jad.tools.componenttools.all_are_contiguous_components_in_same_score() jad.tools.componenttools.copy_components_and_covered_spanne
 281 287
 componenttools.all_are_contiguous_components_in_same_thread() componenttools.copy_components_and_fracture_crossing_spanners()
 (in module ab- (in module ab-
 jad.tools.componenttools.all_are_contiguous_components_in_same_thread() jad.tools.componenttools.copy_components_and_fracture_crossin
 282 289
 componenttools.all_are_orphan_components() componenttools.copy_components_and_immediate_parent_of_first_compo
 (in module ab- (in module ab-
 jad.tools.componenttools.all_are_orphan_components), jad.tools.componenttools.copy_components_and_immediate_pare
 282 290
 componenttools.all_are_thread_contiguous_components() componenttools.copy_components_and_remove_spanners()
 (in module ab- (in module ab-
 jad.tools.componenttools.all_are_thread_contiguous_components() jad.tools.componenttools.copy_components_and_remove_spanne
 282 291
 componenttools.Component (class in ab- componenttools.copy_governed_component_subtree_by_leaf_range()
 jad.tools.componenttools.Component.Component), (in module ab-
 276 jad.tools.componenttools.copy_governed_component_subtree_by
 componenttools.component_to_containment_signature() 292
 (in module ab- componenttools.copy_governed_component_subtree_from_offset_to()
 jad.tools.componenttools.component_to_containment_signature(), module ab-
 283 jad.tools.componenttools.copy_governed_component_subtree_fro
 componenttools.component_to_parentage_signature() 293
 (in module ab- componenttools.cut_component_at_prolated_duration()
 jad.tools.componenttools.component_to_parentage_signature(), (in module ab-
 283 jad.tools.componenttools.cut_component_at_prolated_duration()),
 componenttools.component_to_pitch_and_rhythm_skeleton() 295
 (in module ab- componenttools.extend_in_parent_of_component()
 jad.tools.componenttools.component_to_pitch_and_rhythm(skeleton), module ab-
 284 jad.tools.componenttools.extend_in_parent_of_component()),
 componenttools.component_to_score_depth() 296
 (in module ab- componenttools.extend_left_in_parent_of_component()
 jad.tools.componenttools.component_to_score_depth(), (in module ab-
 285 jad.tools.componenttools.extend_left_in_parent_of_component()),
 componenttools.component_to_score_index() 297
 (in module ab- componenttools.get_component_in_expr_with_name()
 jad.tools.componenttools.component_to_score_index), (in module ab-
 285 jad.tools.componenttools.get_component_in_expr_with_name),
 componenttools.component_to_score_root() 297
 (in module ab- componenttools.get_component_start_offset()
 jad.tools.componenttools.component_to_score_root), (in module ab-
 286 jad.tools.componenttools.get_component_start_offset),

298	(in module ab-	
componenttools.get_component_start_offset_in_seconds()	jad.tools.componenttools.get_improper_descendents_of_component()	
(in module ab-	305	
jad.tools.componenttools.get_component_start_offset_in_seconds()	componenttools.get_improper_parentage_of_component()	
299	(in module ab-	
componenttools.get_component_stop_offset()	jad.tools.componenttools.get_improper_parentage_of_component()	
(in module ab-	305	
jad.tools.componenttools.get_component_stop_offset()	componenttools.get_improper_parentage_of_component_that_start_with_component()	
299	(in module ab-	
componenttools.get_component_stop_offset_in_seconds()	jad.tools.componenttools.get_improper_parentage_of_component()	
(in module ab-	305	
jad.tools.componenttools.get_component_stop_offset_in_seconds()	componenttools.get_improper_parentage_of_component_that_stop_with_component()	
299	(in module ab-	
componenttools.get_components_in_expr_with_name()	jad.tools.componenttools.get_improper_parentage_of_component()	
(in module ab-	306	
jad.tools.componenttools.get_components_in_expr_with_name()	componenttools.get_leftmost_components_with_prolated_duration_at_most_duration()	
300	(in module ab-	
componenttools.get_first_component_in_expr_with_name()	jad.tools.componenttools.get_leftmost_components_with_prolated_duration_at_most_duration()	
(in module ab-	306	
jad.tools.componenttools.get_first_component_in_expr_with_name()	componenttools.get_likely_multiplier_of_components()	
300	(in module ab-	
componenttools.get_first_component_with_name_in_improper_parentage_of_component()	jad.tools.componenttools.get_likely_multiplier_of_components(),	
(in module ab-	307	
jad.tools.componenttools.get_first_component_with_name_in_improper_parentage_of_component()	componenttools.get_lineage_of_component()	
301	(in module ab-	
componenttools.get_first_component_with_name_in_proper_parentage_of_component()	jad.tools.componenttools.get_lineage_of_component()	
(in module ab-	308	
jad.tools.componenttools.get_first_component_with_name_in_proper_parentage_of_component()	componenttools.get_lineage_of_component_that_start_with_component()	
301	(in module ab-	
componenttools.get_first_instance_of_class_in_improper_parentage_of_component()	jad.tools.componenttools.get_lineage_of_component_that_start_with_component()	
(in module ab-	308	
jad.tools.componenttools.get_first_instance_of_class_in_improper_parentage_of_component()	componenttools.get_lineage_of_component_that_stop_with_component()	
302	(in module ab-	
componenttools.get_first_instance_of_class_in_proper_parentage_of_component()	jad.tools.componenttools.get_lineage_of_component_that_stop_with_component()	
(in module ab-	309	
jad.tools.componenttools.get_first_instance_of_class_in_proper_parentage_of_component()	componenttools.get_most_distant_sequential_container_in_improper_parentage_of_component()	
302	(in module ab-	
componenttools.get_improper_contents_of_component()	jad.tools.componenttools.get_most_distant_sequential_container_in_improper_parentage_of_component()	
(in module ab-	309	
jad.tools.componenttools.get_improper_contents_of_component()	componenttools.get_nth_component_in_expr()	
302	(in module ab-	
componenttools.get_improper_descendents_of_component()	jad.tools.componenttools.get_nth_component_in_expr(),	
(in module ab-	310	
jad.tools.componenttools.get_improper_descendents_of_component()	componenttools.get_nth_component_in_time_order_from_component()	
303	(in module ab-	
componenttools.get_improper_descendents_of_component_that_crossed_from_component()	jad.tools.componenttools.get_nth_component_in_time_order_from_component()	
(in module ab-	311	
jad.tools.componenttools.get_improper_descendents_of_component_that_crossed_from_component()	componenttools.get_nth_sibling_from_component()	
303	(in module ab-	
componenttools.get_improper_descendents_of_component_that_start_with_component()	jad.tools.componenttools.get_nth_sibling_from_component(),	
(in module ab-	312	
jad.tools.componenttools.get_improper_descendents_of_component_that_start_with_component()	componenttools.get_nth_sibling_from_component()	
304	(in module ab-	
componenttools.get_improper_descendents_of_component_that_stop_with_component()	jad.tools.componenttools.get_nth_sibling_from_component(),	

312 jad.tools.componenttools.partition_components_by_durations_ge()
componenttools.get_parent_and_start_stop_indices_of_components() 318
(in module ab- componenttools.partition_components_by_durations_le()
jad.tools.componenttools.get_parent_and_start_stop_indices_of_components(), module ab-
313 jad.tools.componenttools.partition_components_by_durations_le()
componenttools.get_proper_contents_of_component() 319
(in module ab- componenttools.remove_component_subtree_from_score_and_spanners()
jad.tools.componenttools.get_proper_contents_of_component(), module ab-
314 jad.tools.componenttools.remove_component_subtree_from_score_and_spanners()
componenttools.get_proper_descendents_of_component() 319
(in module ab- componenttools.replace_components_with_children_of_components()
jad.tools.componenttools.get_proper_descendents_of_component(), module ab-
314 jad.tools.componenttools.replace_components_with_children_of_components()
componenttools.get_proper_parentage_of_component() 321
(in module ab- componenttools.report_component_format_contributions()
jad.tools.componenttools.get_proper_parentage_of_component(), module ab-
315 jad.tools.componenttools.report_component_format_contributions()
componenttools.is_immediate_temporal_successor_of_component() 322
(in module ab- componenttools.split_component_at_offset()
jad.tools.componenttools.is_immediate_temporal_successor_of_component(), module ab-
315 jad.tools.componenttools.split_component_at_offset(),
componenttools.is_orphan_component() (in module ab- 322
jad.tools.componenttools.is_orphan_component), componenttools.split_components_at_offsets()
315 (in module ab-
componenttools.is_well_formed_component() jad.tools.componenttools.split_components_at_offsets(),
(in module ab- 324
jad.tools.componenttools.is_well_formed_component), componenttools.sum_duration_of_components_in_seconds()
315 (in module ab-
componenttools.list_badly_formed_components_in_expr() jad.tools.componenttools.sum_duration_of_components_in_seconds()
(in module ab- 328
jad.tools.componenttools.list_badly_formed_components_in_expr), componenttools.sum_preprolated_duration_of_components()
316 (in module ab-
componenttools.move_component_subtree_to_right_in_immediate_parent_component() 329
(in module ab-
jad.tools.componenttools.move_component_subtree_to_right_in_immediate_parent_component(), componenttools.sum_preprolated_duration_of_components()
316 (in module ab-
componenttools.move_parentage_and_spanners_from_components_to_immediate_parents() 329
(in module ab-
jad.tools.componenttools.move_parentage_and_spanners_from_components_to_immediate_parents), componenttools.sum_preprolated_duration_of_components()
317 (in module ab-
componenttools.number_is_between_start_and_stop_offsets_of_components() 330
(in module ab-
jad.tools.componenttools.number_is_between_start_and_stop_offsets_of_components), componenttools.tabulate_well_formedness_violations_in_expr()
317 (in module ab-
componenttools.number_is_between_start_and_stop_offsets_of_components_in_seconds() 330
(in module ab-
jad.tools.componenttools.number_is_between_start_and_stop_offsets_of_components_in_seconds), componenttools.yield_components_grouped_by_preprolated_duration()
318 (in module ab-
componenttools.partition_components_by_durations_exactly() jad.tools.componenttools.yield_components_grouped_by_preprolated_duration()
(in module ab- 331
jad.tools.componenttools.partition_components_by_durations_exactly), componenttools.yield_groups_of_mixed_klasses_in_sequence()
318 (in module ab-
componenttools.partition_components_by_durations_ge() jad.tools.componenttools.yield_groups_of_mixed_klasses_in_sequence()
(in module ab- 332

`componenttools.yield_topmost_components_grouped_by_type()` (method), 450
 (in module `abjad.tools.componenttools`), 333
`jad.tools.componenttools.yield_topmost_components_grouped_by_type()`, 333
`componenttools.yield_topmost_components_of_class_grouped_by_type()` (method), 334
 (in module `abjad.tools.componenttools`), 334
`jad.tools.componenttools.yield_topmost_components_of_class_grouped_by_type()`, 334
`composite_tree` (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary` attribute), 1724
`configurationtools.AbjadConfig` (class in `abjad.tools.configurationtools`), 1895
`jad.tools.configurationtools.AbjadConfig.AbjadConfig`, 1895
`configurationtools.get_abjad_revision_string()` (in module `abjad.tools.configurationtools`), 1897
`jad.tools.configurationtools.get_abjad_revision_string()`, 1897
`configurationtools.get_abjad_startup_string()` (in module `abjad.tools.configurationtools`), 1898
`jad.tools.configurationtools.get_abjad_startup_string()`, 1898
`configurationtools.get_abjad_version_string()` (in module `abjad.tools.configurationtools`), 1898
`jad.tools.configurationtools.get_abjad_version_string()`, 1898
`configurationtools.get_lilypond_version_string()` (in module `abjad.tools.configurationtools`), 1898
`jad.tools.configurationtools.get_lilypond_version_string()`, 1898
`configurationtools.get_python_version_string()` (in module `abjad.tools.configurationtools`), 1898
`jad.tools.configurationtools.get_python_version_string()`, 1898
`configurationtools.get_tab_width()` (in module `abjad.tools.configurationtools`), 1898
`jad.tools.configurationtools.get_tab_width()`, 1898
`configurationtools.get_text_editor()` (in module `abjad.tools.configurationtools`), 1899
`jad.tools.configurationtools.get_text_editor()`, 1899
`configurationtools.list_abjad_environment_variables()` (in module `abjad.tools.configurationtools`), 1899
`jad.tools.configurationtools.list_abjad_environment_variables()`, 1899
`configurationtools.list_package_dependency_versions()` (in module `abjad.tools.configurationtools`), 1899
`jad.tools.configurationtools.list_package_dependency_versions()`, 1899
`configurationtools.read_abjad_user_config_file()` (in module `abjad.tools.configurationtools`), 1899
`jad.tools.configurationtools.read_abjad_user_config_file()`, 1899
`conjugate()` (`abjad.tools.durationtools.Duration.Duration` method), 446
`conjugate()` (`abjad.tools.durationtools.Offset.Offset` method), 446

361
 containertools.get_first_element_starting_at_or_after_offset() (in module abjad.tools.containertools), 370
 (in module abjad.tools.containertools), 371
 361
 containertools.get_first_element_starting_before_or_at_offset() (in module abjad.tools.containertools), 371
 (in module abjad.tools.containertools), 371
 362
 containertools.get_first_element_starting_strictly_after_offset() (in module abjad.tools.containertools), 371
 (in module abjad.tools.containertools), 371
 362
 containertools.get_first_element_starting_strictly_before_offset() (in module abjad.tools.containertools), 371
 (in module abjad.tools.containertools), 371
 362
 containertools.insert_component() (in module abjad.tools.containertools), 342
 (in module abjad.tools.containertools), 342
 362
 containertools.move_parentage_children_and_spanners_from_one_container_to_another() (in module abjad.tools.containertools), 384
 (in module abjad.tools.containertools), 384
 363
 containertools.remove_leafless_containers_in_expr() (in module abjad.tools.containertools), 978
 (in module abjad.tools.containertools), 978
 364
 containertools.repeat_contents_of_container() (in module abjad.tools.containertools), 1000
 (in module abjad.tools.containertools), 1000
 365
 containertools.repeat_last_n_elements_of_container() (in module abjad.tools.containertools), 1369
 (in module abjad.tools.containertools), 1369
 366
 containertools.replace_container_slice_with_rests() (in module abjad.tools.containertools), 1394
 (in module abjad.tools.containertools), 1394
 366
 containertools.replace_contents_of_target_container_with_contents_of_source_container() (in module abjad.tools.containertools), 1654
 (in module abjad.tools.containertools), 1654
 367
 containertools.report_container_modifications() (in module abjad.tools.containertools), 1829
 (in module abjad.tools.containertools), 1829
 368
 containertools.reverse_contents_of_container() (in module abjad.tools.containertools), 1865
 (in module abjad.tools.containertools), 1865
 369
 containertools.scale_contents_of_container() (in module abjad.tools.containertools), 873
 (in module abjad.tools.containertools), 873

context_name (abjad.tools.contexttools.Context.Context.Context (in module abjad.tools.contexttools.detach_staff_change_marks_attached_to_component), 386

context_name (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock (in module abjad.tools.contexttools.detach_tempo_marks_attached_to_component()), 870

context_name (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff (in module abjad.tools.contexttools.detach_tempo_marks_attached_to_component()), 1371

context_name (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff (in module abjad.tools.contexttools.detach_time_signature_marks_attached_to_component()), 1388

context_name (abjad.tools.scoretools.Score.Score.Score (in module abjad.tools.contexttools.detach_time_signature_marks_attached_to_component()), 1396

context_name (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup (in module abjad.tools.contexttools.DynamicMark (class in abjad.tools.contexttools.DynamicMark.DynamicMark)), 1405

context_name (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff (in module abjad.tools.contexttools.get_clef_mark_attached_to_component()), 1656

context_name (abjad.tools.stafftools.Staff.Staff.Staff (in module abjad.tools.contexttools.get_clef_mark_attached_to_component()), 1665

context_name (abjad.tools.voicetools.Voice.Voice.Voice (in module abjad.tools.contexttools.get_clef_marks_attached_to_component()), 1867

contexts (abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock.LayoutBlock (in module abjad.tools.contexttools.get_clef_marks_attached_to_component()), 874

contexttools.all_are_contexts() (in module abjad.tools.contexttools.all_are_contexts()), 422

contexttools.ClefMark (class in abjad.tools.contexttools.ClefMark.ClefMark), 377

contexttools.ClefMarkInventory (class in abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory), 380

contexttools.Context (class in abjad.tools.contexttools.Context.Context), 384

contexttools.ContextMark (class in abjad.tools.contexttools.ContextMark.ContextMark), 392

contexttools.detach_clef_marks_attached_to_component() (in module abjad.tools.contexttools.detach_clef_marks_attached_to_component()), 422

contexttools.detach_context_marks_attached_to_component() (in module abjad.tools.contexttools.detach_context_marks_attached_to_component()), 423

contexttools.detach_dynamic_marks_attached_to_component() (in module abjad.tools.contexttools.detach_dynamic_marks_attached_to_component()), 429

contexttools.detach_instrument_marks_attached_to_component() (in module abjad.tools.contexttools.detach_instrument_marks_attached_to_component()), 424

contexttools.detach_key_signature_marks_attached_to_component() (in module abjad.tools.contexttools.detach_key_signature_marks_attached_to_component()), 424

contexttools.detach_staff_change_marks_attached_to_component() (in module abjad.tools.contexttools.detach_staff_change_marks_attached_to_component()), 431

contexttools.get_clef_mark_attached_to_component() (in module abjad.tools.contexttools.get_clef_marks_attached_to_component()), 427

contexttools.get_clef_marks_attached_to_component() (in module abjad.tools.contexttools.get_clef_marks_attached_to_component()), 427

contexttools.get_context_mark_attached_to_component() (in module abjad.tools.contexttools.get_context_marks_attached_to_any_improper_parent_of_component()), 428

contexttools.get_context_marks_attached_to_any_improper_parent_of_component() (in module abjad.tools.contexttools.get_context_marks_attached_to_any_improper_parent_of_component()), 428

contexttools.get_context_marks_attached_to_component() (in module abjad.tools.contexttools.get_context_marks_attached_to_component()), 428

contexttools.get_dynamic_mark_attached_to_component() (in module abjad.tools.contexttools.get_dynamic_marks_attached_to_component()), 429

contexttools.get_dynamic_marks_attached_to_component() (in module abjad.tools.contexttools.get_dynamic_marks_attached_to_component()), 429

contexttools.get_effective_clef() (in module abjad.tools.contexttools.get_effective_clef()), 430

contexttools.get_effective_context_mark() (in module abjad.tools.contexttools.get_effective_context_mark()), 430

contexttools.get_effective_dynamic() (in module abjad.tools.contexttools.get_effective_dynamic()), 431

contexttools.get_effective_instrument() (in module abjad.tools.contexttools.get_effective_instrument()), 431

contexttools.get_effective_key_signature() (in module abjad.tools.contexttools.get_effective_key_signature), 432

contexttools.get_effective_staff() (in module abjad.tools.contexttools.get_effective_staff), 432

contexttools.get_effective_tempo() (in module abjad.tools.contexttools.get_effective_tempo), 433

contexttools.get_effective_time_signature() (in module abjad.tools.contexttools.get_effective_time_signature), 433

contexttools.get_instrument_mark_attached_to_component() (in module abjad.tools.contexttools.get_instrument_mark_attached_to_component), 434

contexttools.get_instrument_marks_attached_to_component() (in module abjad.tools.contexttools.get_instrument_marks_attached_to_component), 434

contexttools.get_key_signature_mark_attached_to_component() (in module abjad.tools.contexttools.get_key_signature_mark_attached_to_component), 435

contexttools.get_key_signature_marks_attached_to_component() (in module abjad.tools.contexttools.get_key_signature_marks_attached_to_component), 435

contexttools.get_staff_change_mark_attached_to_component() (in module abjad.tools.contexttools.get_staff_change_mark_attached_to_component), 436

contexttools.get_staff_change_marks_attached_to_component() (in module abjad.tools.contexttools.get_staff_change_marks_attached_to_component), 436

contexttools.get_tempo_mark_attached_to_component() (in module abjad.tools.contexttools.get_tempo_mark_attached_to_component), 437

contexttools.get_tempo_marks_attached_to_component() (in module abjad.tools.contexttools.get_tempo_marks_attached_to_component), 437

contexttools.get_time_signature_mark_attached_to_component() (in module abjad.tools.contexttools.get_time_signature_mark_attached_to_component), 438

contexttools.get_time_signature_marks_attached_to_component() (in module abjad.tools.contexttools.get_time_signature_marks_attached_to_component), 438

contexttools.InstrumentMark (class in abjad.tools.contexttools.InstrumentMark.InstrumentMark), 398

contexttools.is_component_with_clef_mark_attached() (in module abjad.tools.contexttools.is_component_with_clef_mark_attached), 439

contexttools.is_component_with_context_mark_attached() (in module abjad.tools.contexttools.is_component_with_context_mark_attached), 439

contexttools.is_component_with_dynamic_mark_attached() (in module abjad.tools.contexttools.is_component_with_dynamic_mark_attached), 440

contexttools.is_component_with_instrument_mark_attached() (in module abjad.tools.contexttools.is_component_with_instrument_mark_attached), 440

contexttools.is_component_with_key_signature_mark_attached() (in module abjad.tools.contexttools.is_component_with_key_signature_mark_attached), 441

contexttools.is_component_with_staff_change_mark_attached() (in module abjad.tools.contexttools.is_component_with_staff_change_mark_attached), 441

contexttools.is_component_with_tempo_mark_attached() (in module abjad.tools.contexttools.is_component_with_tempo_mark_attached), 442

contexttools.is_component_with_time_signature_mark_attached() (in module abjad.tools.contexttools.is_component_with_time_signature_mark_attached), 442

contexttools.KeySignatureMark (class in abjad.tools.contexttools.KeySignatureMark.KeySignatureMark), 402

contexttools.list_clef_names() (in module abjad.tools.contexttools.list_clef_names), 443

contexttools.set_accidental_style_on_sequential_contexts_in_expr() (in module abjad.tools.contexttools.set_accidental_style_on_sequential_contexts_in_expr), 443

contexttools.StaffChangeMark (class in abjad.tools.contexttools.StaffChangeMark.StaffChangeMark), 406

contexttools.TempoMark (class in abjad.tools.contexttools.TempoMark.TempoMark), 410

contexttools.TempoMarkInventory (class in abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory), 414

contexttools.TimeSignatureMark (class in abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark), 414

jad.tools.contexttools.TimeSignatureMark.TimeSignatureMark() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.
 418 method), 1057
 copy() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ClefMarkInventory.ClefMarkInventory.Clef
 method), 1903 method), 381
 copy() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.ClefMarkInventory.TempoMarkInventory.TempoMarkInventory
 method), 1966 method), 415
 copy() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassSet.HarmonicChromaticIntervalClassSet.ObjectVector.ObjectVector.ObjectChromaticObj
 method), 1135 method), 1905
 copy() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet
 method), 1140 method), 659
 copy() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet
 method), 1149 method), 863
 copy() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet
 method), 1155 method), 866
 copy() (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet
 method), 1089 method), 876
 copy() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet
 method), 1097 method), 859
 copy() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet
 method), 1162 method), 884
 copy() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet
 method), 1165 method), 937
 copy() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet
 method), 1172 method), 954
 copy() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSet.MelodicChromaticIntervalClassSet.MelodicChromaticIntervalClassSet.MelodicChromaticIntervalClassSet
 method), 1181 method), 1138
 copy() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet
 method), 1186 method), 1153
 copy() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet
 method), 1200 method), 1088
 copy() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment
 method), 1212 method), 1095
 copy() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet
 method), 1219 method), 1159
 copy() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector
 method), 1222 method), 1170
 copy() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment
 method), 1240 method), 1178
 copy() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector
 method), 1244 method), 1184
 copy() (abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSetcount() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment
 method), 1111 method), 1198
 copy() (abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVectorcount() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment
 method), 1114 method), 1210
 copy() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet
 method), 1120 method), 1216
 copy() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet.PitchObjectSet
 method), 1127 method), 1237
 copy() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary
 method), 1726 method), 1109
 copy() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.ChordClass
 method), 1804 method), 1251
 copy_subarray() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.PitchArray
 method), 1051 method), 1257

count() (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment.PitchClassObjectSegment (ab-
method), 1118
count() (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment.PitchObjectSegment (ab-
method), 1125
count() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory (ab-
method), 1263
count() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow (ab-
method), 1266
count() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory (ab-
method), 1382
count() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList (ab-
method), 1418
count() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple (ab-
method), 1432
count() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator (ab-
method), 1808
count() (abjad.tools.tonalitytools.Scale.Scale.Scale (ab-
method), 1817
create_named_chromatic_pitch_set_in_pitch_range() (abjad.tools.tonalitytools.Scale.Scale.Scale (ab-
method), 1817
cyclic_nodes (abjad.tools.datastructuretools.Digraph.Digraph.Digraph (ab-
attribute), 1900
D
data (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter (ab-
attribute), 1959
datastructuretools.Digraph (class in abjad.tools.datastructuretools.Digraph.Digraph),
1900
datastructuretools.ImmutableDictionary (class in abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary),
1902
datastructuretools.ObjectInventory (class in abjad.tools.datastructuretools.ObjectInventory.ObjectInventory),
1905
datastructuretools.OrdinalConstant (class in abjad.tools.datastructuretools.OrdinalConstant.OrdinalConstant),
1908
debug (abjad.tools.abctools.Parser.Parser.Parser (ab-
tribute), 1879
debug (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser (ab-
tribute), 2046
debug (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser (ab-
tribute), 2069
debug (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser (ab-
tribute), 2074
debug (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser (ab-
tribute), 1344
decoratortools.requires() (in module abjad.tools.decoratortools.requires), 1909
default_instrument_name (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark (ab-
tribute), 398
default_instrument_name (abjad.tools.instrumenttools.Accordion.Accordion.Accordion (ab-
tribute), 487
default_instrument_name (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute (ab-
tribute), 493
default_instrument_name (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone (ab-
tribute), 499
default_instrument_name (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone (ab-
tribute), 505
default_instrument_name (abjad.tools.instrumenttools.BaronSaxophone.BaronSaxophone.BaronSaxophone (ab-
tribute), 517
default_instrument_name (abjad.tools.instrumenttools.BaronVoice.BaronVoice.BaronVoice (ab-
tribute), 523
default_instrument_name (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet (ab-
tribute), 529
default_instrument_name (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute (ab-
tribute), 535
default_instrument_name (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon (ab-
tribute), 559
default_instrument_name (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone (ab-
tribute), 541
default_instrument_name (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone (ab-
tribute), 547
default_instrument_name (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice (ab-
tribute), 553
default_instrument_name (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet (ab-
tribute), 511
default_instrument_name (abjad.tools.instrumenttools.Cello.Cello.Cello (ab-
tribute), 565
default_instrument_name (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA (ab-
tribute), 571
default_instrument_name (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass (ab-
tribute), 577
default_instrument_name (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet (ab-
tribute), 583
default_instrument_name (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute (ab-
tribute), 589

jad.tools.instrumenttools.BaronetSaxophone.BaronetSaxophone.BaronetSaxophone	jad.tools.instrumenttools.BaronetSaxophone.BaronetSaxophone.BaronetSaxophone	jad.tools.instrumenttools.BaronetSaxophone.BaronetSaxophone.BaronetSaxophone
attribute), 517	attribute), 619	attribute), 619
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.BaronetVoice.BaronetVoice.BaronetVoice	jad.tools.instrumenttools.Flute.Flute.Flute	jad.tools.instrumenttools.Flute.Flute.Flute
attribute), 523	attribute), 625	attribute), 625
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet	jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn	jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn
attribute), 529	attribute), 630	attribute), 630
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute	jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel	jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel
attribute), 535	attribute), 636	attribute), 636
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon	jad.tools.instrumenttools.Guitar.Guitar.Guitar	jad.tools.instrumenttools.Guitar.Guitar.Guitar
attribute), 559	attribute), 642	attribute), 642
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone	jad.tools.instrumenttools.Harp.Harp.Harp	jad.tools.instrumenttools.Harp.Harp.Harp
attribute), 541	attribute), 648	attribute), 648
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone	jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord	jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord
attribute), 547	attribute), 654	attribute), 654
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice	jad.tools.instrumenttools.Marimba.Marimba.Marimba	jad.tools.instrumenttools.Marimba.Marimba.Marimba
attribute), 553	attribute), 663	attribute), 663
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet	jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice	jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice
attribute), 511	attribute), 669	attribute), 669
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.Cello.Cello.Cello	jad.tools.instrumenttools.Oboe.Oboe.Oboe	jad.tools.instrumenttools.Oboe.Oboe.Oboe
attribute), 565	attribute), 675	attribute), 675
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA	jad.tools.instrumenttools.Piano.Piano.Piano	jad.tools.instrumenttools.Piano.Piano.Piano
attribute), 571	attribute), 681	attribute), 681
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass	jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo	jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo
attribute), 577	attribute), 687	attribute), 687
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet	jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone	jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone
attribute), 583	attribute), 693	attribute), 693
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute	jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone	jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone
attribute), 589	attribute), 699	attribute), 699
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon	jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice	jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice
attribute), 601	attribute), 705	attribute), 705
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone	jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone	jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone
attribute), 595	attribute), 711	attribute), 711
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice	jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone	jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone
attribute), 607	attribute), 717	attribute), 717
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name
jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet	jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice	jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice
attribute), 613	attribute), 723	attribute), 723
default_short_instrument_name	(ab- default_short_instrument_name	(ab- default_short_instrument_name

	jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet (ab-attribute), 729	depthwise_inventory (ab-attribute), 1318
default_short_instrument_name (ab-attribute), 735	jad.tools.instrumenttools.Tuba.Tuba.Tuba (ab-attribute), 735	detach() (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark method), 379
default_short_instrument_name (ab-attribute), 741	jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion (ab-attribute), 741	detach() (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark method), 393
default_short_instrument_name (ab-attribute), 747	jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone (ab-attribute), 747	detach() (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark method), 397
default_short_instrument_name (ab-attribute), 753	jad.tools.instrumenttools.Viola.Viola.Viola (ab-attribute), 753	detach() (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark method), 401
default_short_instrument_name (ab-attribute), 759	jad.tools.instrumenttools.Violin.Violin.Violin (ab-attribute), 759	detach() (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark method), 404
default_short_instrument_name (ab-attribute), 765	jad.tools.instrumenttools.Xylophone.Xylophone.Xylophonemethod), 421	detach() (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark method), 408
denominator (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark attribute), 420	jad.tools.instrumenttools.Xylophone.Xylophone.Xylophonemethod), 421	detach() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method), 413
denominator (abjad.tools.durationtools.Duration.Duration.Duration attribute), 446	denominator (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark attribute), 420	detach() (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark method), 421
denominator (abjad.tools.durationtools.Offset.Offset.Offset attribute), 449	denominator (abjad.tools.durationtools.Duration.Duration.Duration attribute), 446	detach() (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 478
denominator (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction attribute), 948	denominator (abjad.tools.durationtools.Offset.Offset.Offset attribute), 449	detach() (abjad.tools.instrumenttools.Accordion.Accordion.Accordion method), 490
denominator (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 982	denominator (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction attribute), 948	detach() (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method), 496
denominator (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 994	denominator (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 982	detach() (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 502
depth (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1050	denominator (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 994	detach() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method), 508
depth (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn attribute), 1055	depth (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1050	detach() (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone method), 526
depth (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 1057	depth (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn attribute), 1055	detach() (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice method), 532
depth (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer attribute), 1324	depth (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 1057	detach() (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method), 538
depth (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1338	depth (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer attribute), 1324	detach() (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method), 544
depth (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode attribute), 1317	depth (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1338	detach() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 558
depth (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1424	depth (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode attribute), 1317	detach() (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method), 564
depth (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1437	depth (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1424	detach() (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method), 570
depthwise_inventory (ab-attribute), 1325	depth (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1437	detach() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 556
depthwise_inventory (ab-attribute), 1338	depthwise_inventory (ab-attribute), 1325	detach() (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method), 514
	jad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer (ab-attribute), 1325	detach() (abjad.tools.instrumenttools.Cello.Cello.Cello method), 568
	jad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf (ab-attribute), 1338	detach() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 574
		detach() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 574

method), 586

detach() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 592

detach() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 604

detach() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 598

detach() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 610

detach() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 616

detach() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 622

detach() (abjad.tools.instrumenttools.Flute.Flute.Flute method), 628

detach() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 633

detach() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 639

detach() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 645

detach() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 651

detach() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 657

detach() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 666

detach() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 672

detach() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 678

detach() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 684

detach() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 690

detach() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 696

detach() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 702

detach() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 708

detach() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 714

detach() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 720

detach() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 726

detach() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 732

detach() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 738

detach() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 744

detach() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 750

detach() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 756

detach() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 762

detach() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 768

detach() (abjad.tools.marktools.Annotation.Annotation.Annotation method), 892

detach() (abjad.tools.marktools.Articulation.Articulation.Articulation method), 895

detach() (abjad.tools.marktools.BarLine.BarLine.BarLine method), 898

detach() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 901

detach() (abjad.tools.marktoolsDirectedMarkDirectedMarkDirectedMark method), 889

detach() (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 905

detach() (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method), 908

detach() (abjad.tools.marktools.Mark.Mark.Mark method), 910

detach() (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method), 912

detach() (abjad.tools.markuptools.Markup.Markup.Markup method), 933

MezzoSopranoVoice (class in abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript attribute), 1915

developer_script_classes (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript attribute), 1915

developer_script_program_names (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript attribute), 1915

SopranoVoice (class in abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript attribute), 1914

AbjGrepScript (class in abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript), 1920

BuildApiScript (class in abjad.tools.developerscripttools.BuildApiScript.BuildApiScript), 1922

CleanScript (class in abjad.tools.developerscripttools.CleanScript.CleanScript), 1924

CountLinewidthsScript (class in abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript), 1927

CountToolsScript (class in abjad.tools.developerscripttools.CountToolsScript.CountToolsScript), 1927

difference() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NichromaticPitchSet.interpointIntervalClass
 method), 1219
 difference() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet
 method), 1240
 difference() (abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet.direction_number (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass
 method), 1111
 difference() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet.direction_number (abjad.tools.pitchtools.MelodicIntervalClassObject.MelodicIntervalClassObject.MelodicIntervalClassObject
 method), 1120
 difference() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet.direction_number (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject.MelodicIntervalObject
 method), 1127
 difference() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.direction_number (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval
 method), 1804
 dimensions (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.dimensions (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval
 attribute), 1050
 dimensions (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn.dimensions (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval
 attribute), 1055
 dimensions (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow.dimensions (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject.MelodicIntervalObject
 attribute), 1057
 direction (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner.direction (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass
 attribute), 225
 direction (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner.direction (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass
 attribute), 232
 direction (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.direction (abjad.tools.pitchtools.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass
 attribute), 240
 direction (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.direction (abjad.tools.pitchtools.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass
 attribute), 249
 direction (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.direction (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass
 attribute), 257
 direction (abjad.tools.marktools.Articulation.Articulation.Articulation.direction (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass
 attribute), 894
 direction (abjad.tools.marktoolsDirectedMark.DirectedMark.DirectedMark.direction (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass
 attribute), 888
 direction (abjad.tools.markuptools.Markup.Markup.Markup.direction (abjad.tools.pitchtools.MelodicIntervalClassObject.MelodicIntervalClassObject.MelodicIntervalClassObject
 attribute), 933
 direction (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner.direction (abjad.tools.pitchtools.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass
 attribute), 1528
 direction (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner.direction (abjad.tools.pitchtools.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass
 attribute), 1537
 direction (abjad.tools.spannertoolsDirectedSpanner.DirectedSpanner.DirectedSpanner.direction (abjad.tools.pitchtools.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass
 attribute), 1501
 direction (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner.direction (abjad.tools.pitchtools.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass
 attribute), 1558
 direction (abjad.tools.spannertoolsPhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner.direction (abjad.tools.pitchtools.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass
 attribute), 1593
 direction (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner.direction (abjad.tools.pitchtools.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass
 attribute), 1606
 direction (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner.direction (abjad.tools.pitchtools.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass.MelodicComplexBeamSpannerClass
 attribute), 1686
 direction_number (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval.direction_number (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval
 attribute), 1174
 direction_number (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.direction_number (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass
 attribute), 1176
 direction_number (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.direction_number (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass
 attribute), 1189

1958 duration (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMix
documentationtools.Documenter (class in ab- attribute), 1704
jad.tools.documentationtools.Documenter.Documenter (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.T
1960 attribute), 1711
documentationtools.FunctionCrawler (class in ab- duration (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeInt
jad.tools.documentationtools.FunctionCrawler.FunctionCrawler (attribute), 1724
1962 duration_in_seconds (ab-
documentationtools.FunctionDocumenter (class in ab- jad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner
jad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter (attribute), 224
1963 duration_in_seconds (ab-
documentationtools.InheritanceGraph (class in ab- jad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanne
jad.tools.documentationtools.InheritanceGraph.InheritanceGraph (attribute), 231
1965 duration_in_seconds (ab-
documentationtools.make_ligeti_example_lilypond_file() jad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComp
(in module ab- attribute), 239
jad.tools.documentationtools.make_ligeti_example_lilypond_file() duration_in_seconds (ab-
1972 jad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredCo
documentationtools.make_reference_manual_lilypond_file() attribute), 248
(in module ab- duration_in_seconds (ab-
jad.tools.documentationtools.make_reference_manual_lilypond_file() jad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpann
1972 attribute), 256
documentationtools.make_text_alignment_example_lilypond_file() duration_in_seconds (ab-
(in module ab- jad.tools.chordtools.Chord.Chord.Chord
jad.tools.documentationtools.make_text_alignment_example_lilypond_file() attribute), 266
1973 duration_in_seconds (ab-
documentationtools.ModuleCrawler (class in ab- jad.tools.containertools.Cluster.Cluster.Cluster
jad.tools.documentationtools.ModuleCrawler.ModuleCrawler (attribute), 336
1968 duration_in_seconds (ab-
documentationtools.Pipe (class in ab- jad.tools.containertools.Container.Container.Container
jad.tools.documentationtools.Pipe.Pipe), attribute), 342
1970 duration_in_seconds (ab-
dominant (abjad.tools.tonalitytools.Scale.Scale.Scale at- jad.tools.containertools.FixedDurationContainer.FixedDurationCo
tribute), 1816 attribute), 349
duplicate_pitch_classes (ab- duration_in_seconds (ab-
jad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet.Context
attribute), 1218 attribute), 384
duration (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark seconds (ab-
attribute), 412 jad.tools.gracetools.GraceContainer.GraceContainer.GraceContai
duration (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark
attribute), 418 duration_in_seconds (ab-
duration (abjad.tools.durationtools.TimespanConstant.TimespanConstant.TimespanConstantLeaf.Leaf.Leaf attribute),
attribute), 453 828
duration (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer (ab-
attribute), 1328 jad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.
duration (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf
attribute), 1342 duration_in_seconds (ab-
duration (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode.DynamicMeasure.DynamicMeasure.Dynar
attribute), 1321 attribute), 990
duration (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment (ab-
attribute), 1352 jad.tools.measuretools.Measure.Measure.Measure
duration (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval (attribute), 1001
attribute), 1707 duration_in_seconds (ab-
duration (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.NaturalHa
attribute), 1700 attribute), 1032

duration_in_seconds (ab- jad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.Me
jad.tools.notetools.Note.Note.Note attribute), (ab-
1037 attribute), 1578
duration_in_seconds (ab- jad.tools.spannertools.OctavationSpanner.OctavationSpanner.Octa
jad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest
attribute), 1308
duration_in_seconds (abjad.tools.resttools.Rest.Rest.Rest (ab-
attribute), 1311 jad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.
attribute), 1592
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff jad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.Pia
attribute), 1369 attribute), 1598
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff jad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner
attribute), 1386 attribute), 1605
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.scoretools.Score.Score.Score at- jad.tools.spannertools.Spanner.Spanner.Spanner
tribute), 1395 attribute), 1507
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup jad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.Staff
attribute), 1404 attribute), 1612
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.skiptools.Skip.Skip.Skip attribute), jad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextS
1495 attribute), 1619
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner jad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner
attribute), 1513 attribute), 1625
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner.TrillSpanner
attribute), 1520 attribute), 1631
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner.RhythmicStaff.RhythmicStaff.RhythmicStaff
attribute), 1527 attribute), 1654
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner.RhythmicStaff.RhythmicStaff.RhythmicStaff
attribute), 1536 attribute), 1663
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.spannertoolsDirectedSpannerDirectedSpannerDirectedSpanner.TieChain.TieChain.TieChain
attribute), 1500 attribute), 1682
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner.TieSpanner.TieSpanner.TieSpanner
attribute), 1543 attribute), 1685
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner.FixedDurationTuplet.FixedDurationTuplet.Fi
attribute), 1550 attribute), 1829
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner.Tuplet.Tuplet.Tuplet
attribute), 1557 attribute), 1840
duration_in_seconds (ab- duration_in_seconds (ab-
jad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner.Voice.Voice.Voice at-
tribute), 1565 attribute), 1865
duration_in_seconds (ab- duration_multiplier (ab-
jad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner
attribute), 1571 attribute), 268
duration_in_seconds (ab- duration_multiplier (abjad.tools.leaftools.Leaf.Leaf.Leaf

attribute), 829

duration_multiplier (abjad.tools.durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator(), 1034

duration_multiplier (abjad.tools.durationtools.duration_tokens_to_least_common_denominator(), 1039

duration_multiplier (abjad.tools.durationtools.duration_tokens_to_rationals(), 1309

duration_multiplier (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest module attribute), 1312

duration_multiplier (abjad.tools.resttools.Rest.Rest.Rest attribute), 1312

duration_multiplier (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1496

durations (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 240

durationtools.all_are_duration_tokens() (in module abjad.tools.durationtools.all_are_duration_tokens), 454

durationtools.all_are_durations() (in module abjad.tools.durationtools.all_are_durations), 455

durationtools.assignable_rational_to_dot_count() (in module abjad.tools.durationtools.assignable_rational_to_dot_count), 455

durationtools.assignable_rational_to_lilypond_duration_string() (in module abjad.tools.durationtools.assignable_rational_to_lilypond_duration_string), 456

durationtools.Duration (class in abjad.tools.durationtools.Duration), 444

durationtools.duration_pair_to_prolation_string() (in module abjad.tools.durationtools.duration_pair_to_prolation_string), 456

durationtools.duration_token_to_assignable_duration_pairs() (in module abjad.tools.durationtools.duration_token_to_assignable_duration_pairs), 456

durationtools.duration_token_to_assignable_lilypond_duration_string() (in module abjad.tools.durationtools.duration_token_to_assignable_lilypond_duration_string), 456

durationtools.duration_token_to_duration_pair() (in module abjad.tools.durationtools.duration_token_to_duration_pair), 457

durationtools.duration_token_to_rational() (in module abjad.tools.durationtools.duration_token_to_rational), 457

durationtools.duration_tokens_to_duration_pairs() (in module abjad.tools.durationtools.duration_tokens_to_duration_pairs), 457

durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator() (in module abjad.tools.durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator), 458

durationtools.duration_tokens_to_least_common_denominator() (in module abjad.tools.durationtools.duration_tokens_to_least_common_denominator), 458

durationtools.duration_tokens_to_rationals() (in module abjad.tools.durationtools.duration_tokens_to_rationals), 458

durationtools.group_duration_tokens_by_implied_prolation() (in module abjad.tools.durationtools.group_duration_tokens_by_implied_prolation), 459

durationtools.is_assignable_rational() (in module abjad.tools.durationtools.is_assignable_rational), 459

durationtools.is_binary_rational() (in module abjad.tools.durationtools.is_binary_rational), 459

durationtools.is_duration_pair() (in module abjad.tools.durationtools.is_duration_pair), 460

durationtools.is_duration_token() (in module abjad.tools.durationtools.is_duration_token), 460

durationtools.is_lilypond_duration_name() (in module abjad.tools.durationtools.is_lilypond_duration_name), 460

durationtools.is_lilypond_duration_string() (in module abjad.tools.durationtools.is_lilypond_duration_string), 461

durationtools.is_proper_tuplet_multiplier() (in module abjad.tools.durationtools.is_proper_tuplet_multiplier), 461

durationtools.lilypond_duration_string_to_rational() (in module abjad.tools.durationtools.lilypond_duration_string_to_rational), 462

durationtools.lilypond_duration_string_to_rational_list() (in module abjad.tools.durationtools.lilypond_duration_string_to_rational_list), 462

durationtools.multiply_duration_pair() (in module abjad.tools.durationtools.multiply_duration_pair), 462

durationtools.multiply_duration_pair_and_reduce_factors() (in module abjad.tools.durationtools.multiply_duration_pair_and_reduce_factors), 462

[durationtools.multiply_duration_pair_and_try_to_preserve_numerator\(\)](#)
 (in module abjad.tools.durationtools, 469)
[jad.tools.durationtools.multiply_duration_pair_and_try_to_preserve_numerator\(\)](#)
 (in module abjad.tools.durationtools, 463)
[durationtools.numeric_seconds_to_clock_string\(\)](#)
 (in module abjad.tools.durationtools, 470)
[jad.tools.durationtools.numeric_seconds_to_clock_string\(\)](#)
 (in module abjad.tools.durationtools, 463)
[durationtools.numeric_seconds_to_escaped_clock_string\(\)](#)
 (in module abjad.tools.durationtools, 453)
[jad.tools.durationtools.numeric_seconds_to_escaped_clock_string\(\)](#)
 (in module abjad.tools.durationtools, 463)
[durationtools.Offset](#) (class in abjad.tools.durationtools, 470)
[jad.tools.durationtools.Offset](#) (class in abjad.tools.durationtools, 449)
[durationtools.positive_integer_to_implied_prolation_multiplier\(\)](#)
 (in module abjad.tools.durationtools, 471)
[jad.tools.durationtools.positive_integer_to_implied_prolation_multiplier\(\)](#)
 (in module abjad.tools.durationtools, 463)
[durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator\(\)](#)
 (in module abjad.tools.durationtools, 471)
[jad.tools.durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator\(\)](#)
 (in module abjad.tools.durationtools, 464)
[durationtools.rational_to_duration_pair_with_specified_integer_denominator\(\)](#)
 (in module abjad.tools.durationtools, 471)
[jad.tools.durationtools.rational_to_duration_pair_with_specified_integer_denominator\(\)](#)
 (in module abjad.tools.durationtools, 465)
[durationtools.rational_to_equal_or_greater_assignable_rational\(\)](#)
 (in module abjad.tools.durationtools, 452)
[jad.tools.durationtools.rational_to_equal_or_greater_assignable_rational\(\)](#)
 (in module abjad.tools.durationtools, 466)
[durationtools.rational_to_equal_or_greater_binary_rational\(\)](#)
 (in module abjad.tools.durationtools, 466)
[jad.tools.durationtools.rational_to_equal_or_greater_binary_rational\(\)](#)
 (in module abjad.tools.durationtools, 466)
[durationtools.rational_to_equal_or_lesser_assignable_rational\(\)](#)
 (in module abjad.tools.durationtools, 1701)
[jad.tools.durationtools.rational_to_equal_or_lesser_assignable_rational\(\)](#)
 (in module abjad.tools.durationtools, 467)
[durationtools.rational_to_equal_or_lesser_binary_rational\(\)](#)
 (in module abjad.tools.durationtools, 1725)
[jad.tools.durationtools.rational_to_equal_or_lesser_binary_rational\(\)](#)
 (in module abjad.tools.durationtools, 468)
[durationtools.rational_to_flag_count\(\)](#) (in module abjad.tools.durationtools, 1711)
[jad.tools.durationtools.rational_to_flag_count\(\)](#)
 (in module abjad.tools.durationtools, 468)
[durationtools.rational_to_fraction_string\(\)](#) (in module abjad.tools.durationtools, 1900)
[jad.tools.durationtools.rational_to_fraction_string\(\)](#)
 (in module abjad.tools.durationtools, 468)
[durationtools.rational_to_prolation_string\(\)](#)
 (in module abjad.tools.durationtools, 377)
[jad.tools.durationtools.rational_to_prolation_string\(\)](#)
 (in module abjad.tools.durationtools, 469)
[durationtools.rational_to_proper_fraction\(\)](#)
 (in module abjad.tools.durationtools, 395)

effective_context (abjad.tools.contexttools.InstrumentMark.EffectiveContextMark.abjad.tools.InstrumentTools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 399

effective_context (abjad.tools.contexttools.KeySignatureMark.EffectiveContextMark.abjad.tools.KeySignatureTools.Flute.Flute.Flute attribute), 402

effective_context (abjad.tools.contexttools.StaffChangeMark.EffectiveContextMark.abjad.tools.StaffChangeTools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 407

effective_context (abjad.tools.contexttools.TempoMark.EffectiveContextMark.abjad.tools.TempoTools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 410

effective_context (abjad.tools.contexttools.TimeSignatureMark.EffectiveContextMark.abjad.tools.TimeSignatureTools.Guitar.Guitar.Guitar attribute), 419

effective_context (abjad.tools.instrumenttools.Accordion.EffectiveContextMark.abjad.tools.AccordionTools.Harp.Harp.Harp attribute), 487

effective_context (abjad.tools.instrumenttools.AltoFlute.EffectiveContextMark.abjad.tools.AltoFluteTools.Harpsichord.Harpsichord.Harpsichord attribute), 493

effective_context (abjad.tools.instrumenttools.AltoSaxophone.EffectiveContextMark.abjad.tools.AltoSaxophoneTools.Marimba.Marimba.Marimba attribute), 499

effective_context (abjad.tools.instrumenttools.AltoTrombone.EffectiveContextMark.abjad.tools.AltoTromboneTools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice attribute), 505

effective_context (abjad.tools.instrumenttools.BaronSaxophone.EffectiveContextMark.abjad.tools.BaronSaxophoneTools.Oboe.Oboe.Oboe attribute), 517

effective_context (abjad.tools.instrumenttools.BaronVoice.EffectiveContextMark.abjad.tools.BaronVoiceTools.Piano.Piano.Piano attribute), 523

effective_context (abjad.tools.instrumenttools.BassClarinet.EffectiveContextMark.abjad.tools.BassClarinetTools.Piccolo.Piccolo.Piccolo attribute), 529

effective_context (abjad.tools.instrumenttools.BassFlute.EffectiveContextMark.abjad.tools.BassFluteTools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone attribute), 535

effective_context (abjad.tools.instrumenttools.Bassoon.EffectiveContextMark.abjad.tools.BassoonTools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 559

effective_context (abjad.tools.instrumenttools.BassSaxophone.EffectiveContextMark.abjad.tools.BassSaxophoneTools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 541

effective_context (abjad.tools.instrumenttools.BassTrombone.EffectiveContextMark.abjad.tools.BassTromboneTools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 547

effective_context (abjad.tools.instrumenttools.BassVoice.EffectiveContextMark.abjad.tools.BassVoiceTools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 553

effective_context (abjad.tools.instrumenttools.BFlatClarinet.EffectiveContextMark.abjad.tools.BFlatClarinetTools.TenorVoice.TenorVoice.TenorVoice attribute), 511

effective_context (abjad.tools.instrumenttools.Cello.EffectiveContextMark.abjad.tools.CelloTools.Trumpet.Trumpet.Trumpet attribute), 565

effective_context (abjad.tools.instrumenttools.ClarinetInA.EffectiveContextMark.abjad.tools.ClarinetInATools.Tuba.Tuba.Tuba attribute), 571

effective_context (abjad.tools.instrumenttools.Contrabass.EffectiveContextMark.abjad.tools.ContrabassTools.UntunedPercussion.UntunedPercussion.UntunedPercussion attribute), 577

effective_context (abjad.tools.instrumenttools.ContrabassClarinet.EffectiveContextMark.abjad.tools.ContrabassClarinetTools.Vibraphone.Vibraphone.Vibraphone attribute), 583

effective_context (abjad.tools.instrumenttools.ContrabassFlute.EffectiveContextMark.abjad.tools.ContrabassFluteTools.Viola.Viola.Viola attribute), 589

effective_context (abjad.tools.instrumenttools.Contrabassoon.EffectiveContextMark.abjad.tools.ContrabassoonTools.Violin.Violin.Violin attribute), 601

effective_context (abjad.tools.instrumenttools.ContrabassSaxophone.EffectiveContextMark.abjad.tools.ContrabassSaxophoneTools.Xylophone.Xylophone.Xylophone attribute), 595

effective_context (abjad.tools.instrumenttools.ContraltoVoice.EffectiveContextMark.abjad.tools.ContraltoVoiceTools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 607

effective_context (abjad.tools.instrumenttools.EFlatClarinet.EffectiveContextMark.abjad.tools.EFlatClarinetTools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 613

attribute), 1888

engraver_consists (abjad.tools.contexttools.Context.ContextContext attribute), 384

engraver_consists (abjad.tools.lilypondfiletools.ContextBlock.ContextBlockContextBlock attribute), 869

engraver_consists (abjad.tools.scoretools.GrandStaff.GrandStaffGrandStaff attribute), 1369

engraver_consists (abjad.tools.scoretools.PianoStaff.PianoStaffPianoStaff attribute), 1386

engraver_consists (abjad.tools.scoretools.Score.Score.Score extend() attribute), 1395

engraver_consists (abjad.tools.scoretools.StaffGroup.StaffGroupStaffGroup attribute), 1404

engraver_consists (abjad.tools.stafftools.RhythmicStaff.RhythmicStaffRhythmicStaff attribute), 1654

engraver_consists (abjad.tools.stafftools.Staff.Staff.Staff extend() attribute), 1663

engraver_consists (abjad.tools.voicetools.Voice.Voice.Voice extend() attribute), 1865

engraver_removals (abjad.tools.contexttools.Context.Context.Context attribute), 385

engraver_removals (abjad.tools.lilypondfiletools.ContextBlock.ContextBlockContextBlock attribute), 869

engraver_removals (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1369

engraver_removals (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1386

engraver_removals (abjad.tools.scoretools.Score.Score.Score attribute), 1395

engraver_removals (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1404

engraver_removals (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1655

engraver_removals (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1664

engraver_removals (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1866

executable (abjad.tools.documentationtools.Pipe.Pipe.Pipe attribute), 1970

extend() (abjad.tools.beamtools.BeamSpanner.BeamSpannerBeamSpanner method), 226

extend() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpannerComplexBeamSpanner method), 234

extend() (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpannerDuratedComplexBeamSpanner method), 242

extend() (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpannerMeasuredComplexBeamSpanner method), 251

extend() (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpannerMultipartBeamSpanner method), 258

extend() (abjad.tools.chordtools.Chord.Chord.Chord method), 269

extend() (abjad.tools.containertools.Cluster.Cluster.Cluster method), 338

extend() (abjad.tools.containertools.Container.Container.Container.Container method), 345

extend() (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainerFixedDurationContainer method), 352

extend() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 381

extend() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 415

extend() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1905

extend() (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 479

extend() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 659

extend() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlockBookBlock method), 863

extend() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 866

extend() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 876

extend() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlockNonattributedBlock method), 859

extend() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlockScoreBlock method), 884

extend() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 937

extend() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 984

extend() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 996

extend() (abjad.tools.measuretools.Measure.Measure.MeasureMeasure method), 1006

extend() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1055

extend() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1058

extend() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1251

extend() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1257

extend() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1263

extend() (abjad.tools.pitchtools.RhythmicTreeContainer.RhythmicTreeContainer.RhythmicTreeContainer method), 1330

extend() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1369

extend() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1386

extend() (abjad.tools.scoretools.Score.Score.Score method), 1395

extend() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1404

extend() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1655

extend() (abjad.tools.stafftools.Staff.Staff.Staff method), 1664

extend() (abjad.tools.voicetools.Voice.Voice.Voice method), 1866

method), 1373

extend() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1382

extend() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1390

extend() (abjad.tools.scoretools.Score.Score.Score method), 1398

extend() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1407

extend() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1418

extend() (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner method), 1515

extend() (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner method), 1521

extend() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1530

extend() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1539

extend() (abjad.tools.spannertools.DirectedSpanner.DirectedSpanner.DirectedSpanner method), 1502

extend() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1546

extend() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1552

extend() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1560

extend() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1566

extend() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1573

extend() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method), 1580

extend() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner method), 1587

extend() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1593

extend() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner method), 1600

extend() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method), 1607

extend() (abjad.tools.spannertools.Spanner.Spanner.Spanner method), 1508

extend() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner method), 1614

extend() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner method), 1620

extend() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner method), 1627

extend() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method), 1633

extend() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1658

extend() (abjad.tools.stafftools.Staff.Staff.Staff method), 1667

extend() (abjad.tools.tuplettools.TieSpanner.TieSpanner.TieSpanner method), 1687

extend() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1835

extend() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1844

extend() (abjad.tools.voicetools.Voice.Voice.Voice method), 1869

extend_left() (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner method), 226

extend_right() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 234

extend_left() (abjad.tools.beamtools.DurationBeamSpanner.DurationBeamSpanner.DurationBeamSpanner method), 243

extend_left() (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 251

extend_left() (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 258

extend_left() (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner method), 1515

extend_left() (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner method), 1522

extend_left() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1531

extend_left() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1539

extend_left() (abjad.tools.spannertools.DirectedSpanner.DirectedSpanner.DirectedSpanner method), 1503

extend_left() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1546

extend_left() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1552

extend_left() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1560

extend_left() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1567

extend_left() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1573

extend_left() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method), 1580

extend_left() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner method), 1587

extend_left() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1594

extend_left() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner method), 1601

extend_left() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method), 1607

extend_left() (abjad.tools.spannertools.Spanner.Spanner.Spanner method), 1508

extend_left() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner method), 1614

extend_left() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner method), 1614

extend_left() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner method), 1614

method), 1621
 extend_left() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner method), 1627
 extend_left() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method), 1634
 extend_left() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner method), 1687
 extent (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass attribute), 1803
 extent (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator attribute), 1807
 extent (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction attribute), 1822
 extent_name (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator attribute), 1807
 extent_to_figured_bass_string() (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator method), 1811
 find_intervals_intersecting_or_tangent_to_offset() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1713
 find_intervals_intersecting_or_tangent_to_offset() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1727
 find_intervals_starting_after_offset() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
 find_intervals_starting_and_stopping_within_interval() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1713
 find_intervals_starting_and_stopping_within_interval() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1728
 find_intervals_starting_and_stopping_within_interval() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
 find_intervals_starting_and_stopping_within_interval() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1714
 find_intervals_starting_and_stopping_within_interval() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1729
 find_intervals_starting_at_offset() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
 find_intervals_starting_at_offset() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1714
 find_intervals_starting_at_offset() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1730
 find_intervals_starting_before_offset() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
 find_intervals_starting_before_offset() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1714
 find_intervals_starting_before_offset() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1730
 find_intervals_starting_or_stopping_at_offset() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
 find_intervals_starting_or_stopping_at_offset() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1715
 find_intervals_starting_or_stopping_at_offset() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1731
 find_intervals_starting_within_interval() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702

`find_intervals_starting_within_interval()` (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1715
`find_intervals_starting_within_interval()` (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1732
`find_intervals_stopping_after_offset()` (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
`find_intervals_stopping_after_offset()` (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1716
`find_intervals_stopping_after_offset()` (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1733
`find_intervals_stopping_at_offset()` (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
`find_intervals_stopping_at_offset()` (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1716
`find_intervals_stopping_at_offset()` (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1734
`find_intervals_stopping_before_offset()` (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
`find_intervals_stopping_before_offset()` (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1716
`find_intervals_stopping_before_offset()` (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1735
`find_intervals_stopping_within_interval()` (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
`find_intervals_stopping_within_interval()` (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1717
`find_intervals_stopping_within_interval()` (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1735
`fingered_pitch` (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic attribute), 1033
`fingered_pitch` (abjad.tools.notetools.Note.Note attribute), 1037
`fingered_pitches` (abjad.tools.chordtools.Chord.Chord attribute), 267
`force_fraction` (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1832
`force_fraction` (abjad.tools.tuplettools.Tuplet.Tuplet attribute), 1842
`force_quotes` (abjad.tools.schemetools.Scheme.Scheme attribute), 1348

formatted_help (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.tools.update_script (attribute), 1952
 formatted_usage (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.abjadbooktools.AbjadBookScript (attribute), 1887
 formatted_usage (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript(abjad.tools.developerscripttools.MakeNewClassTemplateScript (attribute), 1915
 formatted_usage (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript(abjad.tools.developerscripttools.MakeNewFunctionTemplateScript (attribute), 1918
 formatted_usage (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript(abjad.tools.developerscripttools.RenameModulesScript (attribute), 1920
 formatted_usage (abjad.tools.developerscripttools.CleanScript.CleanScript(abjad.tools.developerscripttools.ReplaceInFilesScript (attribute), 1923
 formatted_usage (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript(abjad.tools.developerscripttools.ReplacePromptsScript (attribute), 1925
 formatted_usage (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript(abjad.tools.developerscripttools.RunDoctestsScript (attribute), 1927
 formatted_usage (abjad.tools.developerscripttools.DeveloperScript.DeveloperScript(abjad.tools.developerscripttools.SvnAddAllScript (attribute), 1910
 formatted_usage (abjad.tools.developerscripttools.DirectoryScript.DirectoryScript(abjad.tools.developerscripttools.SvnCommitScript (attribute), 1912
 formatted_usage (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript(abjad.tools.developerscripttools.SvnMessageScript (attribute), 1930
 formatted_usage (abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript(abjad.tools.developerscripttools.SvnUpdateScript (attribute), 1932
 formatted_usage (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript(abjad.tools.developerscripttools.ReplaceInFilesScript (attribute), 1934
 formatted_usage (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript(abjad.tools.developerscripttools.ReplacePromptsScript (attribute), 1937
 formatted_usage (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript(abjad.tools.developerscripttools.RunDoctestsScript (attribute), 1940
 formatted_usage (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript(abjad.tools.developerscripttools.SvnAddAllScript (attribute), 1942
 formatted_usage (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript(abjad.tools.developerscripttools.SvnCommitScript (attribute), 1944
 formatted_usage (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript(abjad.tools.developerscripttools.SvnMessageScript (attribute), 1947
 formatted_usage (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript(abjad.tools.developerscripttools.SvnUpdateScript (attribute), 1949
 formatted_usage (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript(abjad.tools.developerscripttools.SvnAddAllScript (attribute), 1952
 formatted_usage (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript (in module abjad.tools.developerscripttools.ReplaceInFilesScript (attribute), 2030
 formatted_usage (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript (in module abjad.tools.developerscripttools.RunDoctestsScript (attribute), 2031
 formatted_usage (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript (in module abjad.tools.developerscripttools.SvnAddAllScript (attribute), 2031
 formatted_usage (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript (in module abjad.tools.developerscripttools.SvnCommitScript (attribute), 2031
 formatted_usage (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript (in module abjad.tools.developerscripttools.SvnMessageScript (attribute), 2031
 formatted_usage (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript (in module abjad.tools.developerscripttools.SvnUpdateScript (attribute), 2031
 formatted_usage (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript (in module abjad.tools.developerscripttools.SvnAddAllScript (attribute), 2031
 formatted_version (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript (in module abjad.tools.abjadbooktools.AbjadBookScript (attribute), 1887
 formatted_version (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript (in module abjad.tools.developerscripttools.AbjDevScript (attribute), 1916
 formatted_version (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript (in module abjad.tools.developerscripttools.AbjGrepScript (attribute), 1918
 formatted_version (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript (in module abjad.tools.developerscripttools.BuildApiScript (attribute), 1920
 formatted_version (abjad.tools.developerscripttools.CleanScript.CleanScript (in module abjad.tools.developerscripttools.CleanScript (attribute), 1923
 formatted_version (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript (in module abjad.tools.developerscripttools.CountLinewidthsScript (attribute), 1925
 formatted_version (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript (in module abjad.tools.developerscripttools.CountToolsScript (attribute), 1927

fuse()	(abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner	method), 1516
fuse()	(abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner	method), 1172
fuse()	(abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner	method), 1522
fuse()	(abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner	method), 1181
fuse()	(abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner	method), 1531
fuse()	(abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner	method), 1222
fuse()	(abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner	method), 1540
fuse()	(abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner	method), 1235
fuse()	(abjad.tools.spannertools.DirectedSpanner.DirectedSpanner.DirectedSpanner	method), 1503
fuse()	(abjad.tools.spannertools.DirectedSpanner.DirectedSpanner.DirectedSpanner	method), 1244
fuse()	(abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner	method), 1547
fuse()	(abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner	method), 1114
fuse()	(abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner	method), 1553
fuse()	(abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner	method), 1708
fuse()	(abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner	method), 1561
fuse()	(abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner	method), 1736
fuse()	(abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner	method), 1567
fuse()	(abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner	method), 1574
fuse()	(abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner	method), 1581
fuse()	(abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner	method), 1492
fuse()	(abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner	method), 1588
fuse()	(abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner	method), 1896
fuse()	(abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner	method), 1594
fuse()	(abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner	method), 1601
fuse()	(abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner	method), 1445
fuse()	(abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner	method), 1608
fuse()	(abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner	method), 1493
fuse()	(abjad.tools.spannertools.Spanner.Spanner.Spanner	method), 1509
fuse()	(abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner	method), 1615
fuse()	(abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner	method), 1621
fuse()	(abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner	method), 1628
fuse()	(abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner	method), 1634
fuse()	(abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner	method), 1688
G		
get()	(abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig	method), 1896
get()	(abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig	method), 520
get()	(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary	method), 1903
get()	(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary	method), 526
get()	(abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph	method), 1966
get()	(abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector	method), 1135
get()	(abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector	method), 532
get()	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector	method), 1165
get()	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector	method), 538

get_default_performer_name() jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 562	(ab- get_default_performer_name() jad.tools.instrumenttools.Guitar.Guitar.Guitar method), 645	(ab-
get_default_performer_name() jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method), 544	(ab- get_default_performer_name() jad.tools.instrumenttools.Harp.Harp.Harp method), 651	(ab-
get_default_performer_name() jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method), 550	(ab- get_default_performer_name() jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 657	(ab-
get_default_performer_name() jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 556	(ab- get_default_performer_name() jad.tools.instrumenttools.Marimba.Marimba.Marimba method), 666	(ab-
get_default_performer_name() jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method), 514	(ab- get_default_performer_name() jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice method), 672	(ab-
get_default_performer_name() jad.tools.instrumenttools.Cello.Cello.Cello method), 568	(ab- get_default_performer_name() jad.tools.instrumenttools.Oboe.Oboe.Oboe method), 678	(ab-
get_default_performer_name() jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 574	(ab- get_default_performer_name() jad.tools.instrumenttools.Piano.Piano.Piano method), 684	(ab-
get_default_performer_name() jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 580	(ab- get_default_performer_name() jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 690	(ab-
get_default_performer_name() jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 586	(ab- get_default_performer_name() jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone method), 696	(ab-
get_default_performer_name() jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 592	(ab- get_default_performer_name() jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone method), 702	(ab-
get_default_performer_name() jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 604	(ab- get_default_performer_name() jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 708	(ab-
get_default_performer_name() jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 598	(ab- get_default_performer_name() jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 714	(ab-
get_default_performer_name() jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 610	(ab- get_default_performer_name() jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 720	(ab-
get_default_performer_name() jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 616	(ab- get_default_performer_name() jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 726	(ab-
get_default_performer_name() jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 622	(ab- get_default_performer_name() jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 732	(ab-
get_default_performer_name() jad.tools.instrumenttools.Flute.Flute.Flute method), 628	(ab- get_default_performer_name() jad.tools.instrumenttools.Tuba.Tuba.Tuba method), 738	(ab-
get_default_performer_name() jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 633	(ab- get_default_performer_name() jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 744	(ab-
get_default_performer_name() jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 639	(ab- get_default_performer_name() jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 750	(ab-

get_default_performer_name() jad.tools.instrumenttools.Viola.Viola.Viola method), 756	(ab- get_performer_names() jad.tools.instrumenttools.Accordion.Accordion.Accordion method), 490	(ab-
get_default_performer_name() jad.tools.instrumenttools.Violin.Violin.Violin method), 762	(ab- get_performer_names() jad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method), 496	(ab-
get_default_performer_name() jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 768	(ab- get_performer_names() jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 502	(ab-
get_initial_comment() jad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1896	(ab- get_performer_names() jad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method), 508	(ab-
get_next_n_complete_nodes_at_level() jad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1426	(ab- get_performer_names() jad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone method), 520	(ab-
get_next_n_complete_nodes_at_level() jad.tools.sequencetools.Tree.Tree.Tree method), 1439	(ab- get_performer_names() jad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice method), 526	(ab-
get_next_n_nodes_at_level() jad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1426	(ab- get_performer_names() jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method), 532	(ab-
get_next_n_nodes_at_level() jad.tools.sequencetools.Tree.Tree.Tree method), 1440	(ab- get_performer_names() jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method), 538	(ab-
get_node_at_position() jad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1427	(ab- get_performer_names() jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 562	(ab-
get_node_at_position() jad.tools.sequencetools.Tree.Tree.Tree method), 1440	(ab- get_performer_names() jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method), 544	(ab-
get_option_comments() jad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1896	(ab- get_performer_names() jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method), 550	(ab-
get_option_definitions() jad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1896	(ab- get_performer_names() jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 556	(ab-
get_option_specs() jad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1896	(ab- get_performer_names() jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method), 514	(ab-
get_overlap_with_interval() jad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1708	(ab- get_performer_names() jad.tools.instrumenttools.Cello.Cello.Cello method), 568	(ab-
get_overlap_with_interval() jad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702	(ab- get_performer_names() jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 574	(ab-
get_overlap_with_interval() jad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin method), 1705	(ab- get_performer_names() jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 580	(ab-
get_overlap_with_interval() jad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1717	(ab- get_performer_names() jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet method), 586	(ab-
get_overlap_with_interval() jad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1736	(ab- get_performer_names() jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute method), 592	(ab-

get_performer_names() jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon. method), 604	(ab- get_performer_names() method), 708	(ab- jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice. method), 708
get_performer_names() jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone. method), 598	(ab- get_performer_names() method), 714	(ab- jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone. method), 714
get_performer_names() jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice. method), 610	(ab- get_performer_names() method), 720	(ab- jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone. method), 720
get_performer_names() jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet. method), 616	(ab- get_performer_names() method), 726	(ab- jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice. method), 726
get_performer_names() jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn. method), 622	(ab- get_performer_names() method), 732	(ab- jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet. method), 732
get_performer_names() jad.tools.instrumenttools.Flute.Flute.Flute method), 628	(ab- get_performer_names() method), 738	(ab- jad.tools.instrumenttools.Tuba.Tuba.Tuba method), 738
get_performer_names() jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn. method), 633	(ab- get_performer_names() method), 744	(ab- jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion. method), 744
get_performer_names() jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel. method), 639	(ab- get_performer_names() method), 750	(ab- jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone. method), 750
get_performer_names() jad.tools.instrumenttools.Guitar.Guitar.Guitar method), 645	(ab- get_performer_names() method), 756	(ab- jad.tools.instrumenttools.Viola.Viola.Viola method), 756
get_performer_names() jad.tools.instrumenttools.Harp.Harp.Harp method), 651	(ab- get_performer_names() method), 762	(ab- jad.tools.instrumenttools.Violin.Violin.Violin method), 762
get_performer_names() jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord. method), 657	(ab- get_performer_names() method), 768	(ab- jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone. method), 768
get_performer_names() jad.tools.instrumenttools.Marimba.Marimba.Marimba method), 666	(ab- get_position_of_descendant() method), 1427	(ab- jad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree. method), 1427
get_performer_names() jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice. method), 672	(ab- get_position_of_descendant() method), 1440	(ab- jad.tools.sequencetools.Tree.Tree.Tree. method), 1440
get_performer_names() jad.tools.instrumenttools.Oboe.Oboe.Oboe method), 678	(ab- global_staff_size (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile. attribute), 875	(ab- global_staff_size (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile. attribute), 875
get_performer_names() jad.tools.instrumenttools.Piano.Piano.Piano method), 684	(ab- governors (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment. attribute), 1859	(ab- governors (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment. attribute), 1859
get_performer_names() jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 690	(ab- grace() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy. method), 2035	(ab- grace() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy. method), 2035
get_performer_names() jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone. method), 696	(ab- gracetools.all_are_grace_containers() (in module ab- jad.tools.gracetools.all_are_grace_containers()), 482	(ab- gracetools.all_are_grace_containers() (in module ab- jad.tools.gracetools.all_are_grace_containers()), 482
get_performer_names() jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone. method), 702	(ab- gracetools.detach_grace_containers_attached_to_leaf() method), 483	(ab- gracetools.detach_grace_containers_attached_to_leaf() method), 483
	(ab- gracetools.detach_grace_containers_attached_to_leaves_in_expr() (in module ab-	(ab- gracetools.detach_grace_containers_attached_to_leaves_in_expr() (in module ab-

jad.tools.gracetools.detach_grace_containers_attached_to_leaf() (ab-
484 attribute), 1140
gracetools.get_grace_containers_attached_to_leaf() (in module ab-
(in module ab- attribute), 1146
jad.tools.gracetools.get_grace_containers_attached_to_leaf() (in module ab-
484 attribute), 1146
gracetools.GraceContainer (class in ab- attribute), 1193
jad.tools.gracetools.GraceContainer.GraceContainer (class in ab-
474 attribute), 1193
graphviz_format (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph
attribute), 1966
harmonic_chromatic_interval (ab- harmonic_diatonic_interval_class (ab-
jad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval (ab-
attribute), 1175 attribute), 1146
harmonic_chromatic_interval (ab- harmonic_diatonic_interval_class_segment (ab-
jad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval (ab-
attribute), 1193 attribute), 1215
harmonic_chromatic_interval_class (ab- harmonic_diatonic_interval_classes (ab-
jad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval (ab-
attribute), 1131 attribute), 1149
harmonic_chromatic_interval_class_segment (ab- harmonic_diatonic_interval_numbers (ab-
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment (ab-
attribute), 1215 attribute), 1155
harmonic_chromatic_interval_numbers (ab- harmonic_diatonic_interval_segment (ab-
jad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet (ab-
attribute), 1140 attribute), 1197
harmonic_chromatic_interval_segment (ab- harmonic_diatonic_interval_segment (ab-
jad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment (ab-
attribute), 1152 attribute), 1215
harmonic_chromatic_interval_segment (ab- harmonic_diatonic_interval_set (ab-
jad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment (ab-
attribute), 1183 attribute), 1200
harmonic_chromatic_interval_segment (ab- harmonic_diatonic_intervals (ab-
jad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment (ab-
attribute), 1197 attribute), 1155
harmonic_chromatic_interval_segment (ab- has_key() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableD
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment (method), 1903
attribute), 1215 has_key() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGra
harmonic_chromatic_interval_segment (ab- method), 1966
jad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator (method), 1135
attribute), 1807 has_key() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.H
harmonic_chromatic_interval_set (ab- has_key() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClas
jad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet (method), 1165
attribute), 1154 has_key() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassV
harmonic_chromatic_interval_set (ab- method), 1172
jad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet (method), 1181
attribute), 1186 has_key() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.Me
harmonic_chromatic_interval_set (ab- has_key() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChrom
jad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet (method), 1228
attribute), 1199 has_key() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.Nur
harmonic_chromatic_intervals (ab- method), 1244
jad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet (method), 1244

method), 1114
has_key() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.
method), 1736
has_none_of() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.
method), 1135
has_spanning_cell_over_index() (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf.
method), 1051
has_spanning_cell_over_index() (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode.
method), 1058
has_voice_crossing (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree.
attribute), 1050
has_voice_crossing (abjad.tools.sequencetools.Tree.Tree.Tree.
attribute), 1055
head (abjad.tools.tietools.TieChain.TieChain.TieChain.
attribute), 1682
hide (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock.
attribute), 1888
HOME_PATH (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig.
attribute), 1896
I
ignored_directories (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler.
attribute), 1968
imag (abjad.tools.durationtools.Duration.Duration.Duration.
attribute), 446
imag (abjad.tools.durationtools.Offset.Offset.Offset.
attribute), 450
imag (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction.
attribute), 948
image_block (abjad.tools.abjadbooktools.HTMLOutputFormat.HTMLOutputFormat.HTMLOutputFormat.
attribute), 1890
image_block (abjad.tools.abjadbooktools.LaTeXOutputFormat.LaTeXOutputFormat.LaTeXOutputFormat.
attribute), 1892
image_block (abjad.tools.abjadbooktools.OutputFormat.OutputFormat.OutputFormat.
attribute), 1884
image_block (abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutputFormat.ReSTOutputFormat.
attribute), 1893
image_format (abjad.tools.abjadbooktools.HTMLOutputFormat.HTMLOutputFormat.HTMLOutputFormat.
attribute), 1890
image_format (abjad.tools.abjadbooktools.LaTeXOutputFormat.LaTeXOutputFormat.LaTeXOutputFormat.
attribute), 1892
image_format (abjad.tools.abjadbooktools.OutputFormat.OutputFormat.OutputFormat.
attribute), 1884
image_format (abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutputFormat.ReSTOutputFormat.
attribute), 1893
image_prefix (abjad.tools.abjadbooktools.AbjadBookProcessor.AbjadBookProcessor.AbjadBookProcessor.
attribute), 1885
importtools.import_structured_package() (in module abjad.tools.importtools.import_structured_package).
method), 415
index() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.
method), 2033
index() (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer.
method), 1335
index() (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf.
attribute), 1338
index() (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode.
attribute), 1318
index() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree.
attribute), 1424
index() (abjad.tools.sequencetools.Tree.Tree.Tree.
attribute), 1437
include_private_objects (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler.
attribute), 1957
include_private_objects (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler.
attribute), 1528
include_private_objects (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner.
attribute), 1528
include_private_objects (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner.
attribute), 1537
include_private_objects (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner.
attribute), 1558
indented_lilypond_format (abjad.tools.markuptools.Markup.Markup.Markup.
attribute), 932
index() (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner.
method), 227
index() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner.
method), 235
index() (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.
method), 244
index() (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.
method), 253
index() (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.
method), 259
index() (abjad.tools.containertools.Cluster.Cluster.Cluster.
method), 338
index() (abjad.tools.containertools.Container.Container.Container.
method), 345
index() (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer.
method), 353
index() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.
method), 381
index() (abjad.tools.contexttools.Context.Context.Context.
method), 388
index() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.
method), 415

- method), 1905
- index() (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 479
- index() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 659
- index() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 863
- index() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 866
- index() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 876
- index() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 859
- index() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 884
- index() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 938
- index() (abjad.tools.mathtools.Ratio.Ratio.Ratio method), 954
- index() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 985
- index() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 996
- index() (abjad.tools.measuretools.Measure.Measure.Measure method), 1006
- index() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1058
- index() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1138
- index() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1153
- index() (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment.IntervalClassObjectSegment method), 1088
- index() (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment.IntervalObjectSegment method), 1095
- index() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSegment.InversionEquivalentChromaticIntervalSegment.InversionEquivalentChromaticIntervalSegment method), 1160
- index() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalSegment.InversionEquivalentDiatonicIntervalSegment.InversionEquivalentDiatonicIntervalSegment method), 1170
- index() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment method), 1179
- index() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment method), 1184
- index() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 1198
- index() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1210
- index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment method), 1216
- index() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1237
- index() (abjad.tools.pitchtools.ObjectSegment.ObjectSegment.ObjectSegment method), 1109
- index() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1252
- index() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1257
- index() (abjad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment.PitchClassObjectSegment method), 1118
- index() (abjad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment.PitchObjectSegment method), 1125
- index() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1263
- index() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 1266
- index() (abjad.tools.pitchtools.NonythirtetupleRhythmTreeContainer.NonythirtetupleRhythmTreeContainer.NonythirtetupleRhythmTreeContainer method), 1331
- index() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1373
- index() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1383
- index() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1390
- index() (abjad.tools.scoretools.Score.Score.Score method), 1399
- index() (abjad.tools.stafftools.StaffGroup.StaffGroup.StaffGroup method), 1408
- index() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1418
- index() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple method), 1432
- index() (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner method), 1516
- index() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1523
- index() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1532
- index() (abjad.tools.spannertools.DuplicateSpanner.DuplicateSpanner.DuplicateSpanner method), 1540
- index() (abjad.tools.spannertools.EquivalentSpanner.EquivalentSpanner.EquivalentSpanner method), 1504
- index() (abjad.tools.spannertools.DynamicSpanner.DynamicSpanner.DynamicSpanner method), 1547
- index() (abjad.tools.spannertools.GlideSpanner.GlideSpanner.GlideSpanner method), 1553
- index() (abjad.tools.spannertools.MelodicChromaticIntervalSpanner.MelodicChromaticIntervalSpanner.MelodicChromaticIntervalSpanner method), 1562
- index() (abjad.tools.spannertools.MelodicDiatonicIntervalSpanner.MelodicDiatonicIntervalSpanner.MelodicDiatonicIntervalSpanner method), 1568
- index() (abjad.tools.spannertools.NakedSpanner.NakedSpanner.NakedSpanner method), 1574
- index() (abjad.tools.spannertools.NamedChromaticPitchSegmentSpanner.NamedChromaticPitchSegmentSpanner.NamedChromaticPitchSegmentSpanner method), 1581
- index() (abjad.tools.spannertools.NumberedChromaticPitchClassSegmentSpanner.NumberedChromaticPitchClassSegmentSpanner.NumberedChromaticPitchClassSegmentSpanner method), 1589
- index() (abjad.tools.spannertools.OctaveTranspositionMappingInventorySpanner.OctaveTranspositionMappingInventorySpanner.OctaveTranspositionMappingInventorySpanner method), 1595
- index() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1595
- index() (abjad.tools.spannertools.PedalSpanner.PedalSpanner.PedalSpanner method), 1595

method), 1602

index() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method), 1608

index() (abjad.tools.spannertools.Spanner.Spanner.Spanner method), 1509

index() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner method), 1615

index() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner method), 1622

index() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner method), 1628

index() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method), 1635

index() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1659

index() (abjad.tools.stafftools.Staff.Staff.Staff method), 1668

index() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner method), 1688

index() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 1808

index() (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1817

index() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1835

index() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1845

index() (abjad.tools.voicetools.Voice.Voice.Voice method), 1870

index_in_parent (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1424

index_in_parent (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1438

indices (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell attribute), 1053

inflection_point_count (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1215

inherited_attributes (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter attribute), 1959

insert() (abjad.tools.containertools.Cluster.Cluster.Cluster method), 339

insert() (abjad.tools.containertools.Container.Container.Container method), 345

insert() (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer method), 353

insert() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 381

insert() (abjad.tools.contexttools.Context.Context.Context method), 389

insert() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 415

insert() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1905

insert() (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 479

insert() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 659

insert() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 863

insert() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 867

insert() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 876

insert() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 859

insert() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 884

insert() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 938

insert() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 985

insert() (abjad.tools.measuretools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 997

insert() (abjad.tools.measuretools.Measure.Measure.Measure method), 1006

insert() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1252

insert() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1257

insert() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1263

insert() (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer method), 1331

insert() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1374

insert() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1383

insert() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1399

insert() (abjad.tools.scoretools.Score.Score.Score method), 1408

insert() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1418

insert() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1659

insert() (abjad.tools.stafftools.FixedDurationStaff.FixedDurationStaff.FixedDurationStaff method), 1668

insert() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1836

insert() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1845

insert() (abjad.tools.voicetools.Voice.Voice.Voice method), 1870

insert() (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier.InstrumentationSpecifier method), 1905

2241

instrument_name_markup jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone. attribute), 501	(ab- instrument_name_markup jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice. attribute), 608	(ab- instrument_name_markup jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice. attribute), 608
instrument_name_markup jad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone. attribute), 507	(ab- instrument_name_markup jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet. attribute), 615	(ab- instrument_name_markup jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet. attribute), 615
instrument_name_markup jad.tools.instrumenttools.BartitoneSaxophone.BartitoneSaxophone.BartitoneSaxophone. attribute), 519	(ab- instrument_name_markup jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn. attribute), 621	(ab- instrument_name_markup jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn. attribute), 621
instrument_name_markup jad.tools.instrumenttools.BartitoneVoice.BartitoneVoice.BartitoneVoice. attribute), 524	(ab- instrument_name_markup jad.tools.instrumenttools.Flute.Flute.Flute attribute), 626	(ab- instrument_name_markup jad.tools.instrumenttools.Flute.Flute.Flute attribute), 626
instrument_name_markup jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet. attribute), 531	(ab- instrument_name_markup jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 632	(ab- instrument_name_markup jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 632
instrument_name_markup jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 537	(ab- instrument_name_markup jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel. attribute), 637	(ab- instrument_name_markup jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel. attribute), 637
instrument_name_markup jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 561	(ab- instrument_name_markup jad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 643	(ab- instrument_name_markup jad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 643
instrument_name_markup jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone. attribute), 543	(ab- instrument_name_markup jad.tools.instrumenttools.Harp.Harp.Harp attribute), 650	(ab- instrument_name_markup jad.tools.instrumenttools.Harp.Harp.Harp attribute), 650
instrument_name_markup jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone. attribute), 549	(ab- instrument_name_markup jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord. attribute), 656	(ab- instrument_name_markup jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord. attribute), 656
instrument_name_markup jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 554	(ab- instrument_name_markup jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 664	(ab- instrument_name_markup jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 664
instrument_name_markup jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet. attribute), 513	(ab- instrument_name_markup jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice. attribute), 670	(ab- instrument_name_markup jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice. attribute), 670
instrument_name_markup jad.tools.instrumenttools.Cello.Cello.Cello attribute), 566	(ab- instrument_name_markup jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 677	(ab- instrument_name_markup jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 677
instrument_name_markup jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA. attribute), 573	(ab- instrument_name_markup jad.tools.instrumenttools.Piano.Piano.Piano attribute), 683	(ab- instrument_name_markup jad.tools.instrumenttools.Piano.Piano.Piano attribute), 683
instrument_name_markup jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass. attribute), 578	(ab- instrument_name_markup jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 689	(ab- instrument_name_markup jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 689
instrument_name_markup jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet. attribute), 585	(ab- instrument_name_markup jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone. attribute), 695	(ab- instrument_name_markup jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone. attribute), 695
instrument_name_markup jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute. attribute), 591	(ab- instrument_name_markup jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone. attribute), 701	(ab- instrument_name_markup jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone. attribute), 701
instrument_name_markup jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon. attribute), 603	(ab- instrument_name_markup jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice. attribute), 706	(ab- instrument_name_markup jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice. attribute), 706
instrument_name_markup jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone. attribute), 597	(ab- instrument_name_markup jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone. attribute), 713	(ab- instrument_name_markup jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone. attribute), 713

instrument_name_markup (ab- 534
jad.tools.instrumenttools.TenorTrombone.TenorTrombone (class in ab-
attribute), 719 jad.tools.instrumenttools.Bassoon.Bassoon),

instrument_name_markup (ab- 558
jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice (class in ab-
attribute), 724 jad.tools.instrumenttools.BassSaxophone.BassSaxophone),

instrument_name_markup (ab- 540
jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet (class in ab-
attribute), 730 jad.tools.instrumenttools.BassTrombone.BassTrombone),

instrument_name_markup (ab- 546
jad.tools.instrumenttools.Tuba.Tuba.Tuba instrumenttools.BassVoice (class in ab-
attribute), 736 jad.tools.instrumenttools.BassVoice.BassVoice),

instrument_name_markup (ab- 552
jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion (class in ab-
attribute), 742 jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet),

instrument_name_markup (ab- 510
jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone (class in ab-
attribute), 748 jad.tools.instrumenttools.Cello.Cello), 564

instrument_name_markup (ab- instrumenttools.ClarineteInA (class in ab-
jad.tools.instrumenttools.Viola.Viola.Viola jad.tools.instrumenttools.ClarineteInA.ClarineteInA),
attribute), 754 570

instrument_name_markup (ab- instrumenttools.Contrabass (class in ab-
jad.tools.instrumenttools.Violin.Violin.Violin jad.tools.instrumenttools.Contrabass.Contrabass),
attribute), 760 576

instrument_name_markup (ab- instrumenttools.ContrabassClarinet (class in ab-
jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet),
attribute), 766 582

instruments (abjad.tools.scoretools.InstrumentationSpecifier instrumenttools.Specifier instrumenttools.Specifier ab-
attribute), 1378 jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute),

instruments (abjad.tools.scoretools.Performer.Performer.Performer 588
attribute), 1381 instrumenttools.Contrabassoon (class in ab-
instrumenttools.Accordion (class in ab- jad.tools.instrumenttools.Contrabassoon.Contrabassoon),
jad.tools.instrumenttools.Accordion.Accordion), 600

instrumenttools.AltoFlute (class in ab- instrumenttools.ContrabassSaxophone (class in ab-
jad.tools.instrumenttools.AltoFlute.AltoFlute), 594
492 instrumenttools.ContraltoVoice (class in ab-
instrumenttools.AltoSaxophone (class in ab- jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice),
jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone), 606
498 instrumenttools.default_instrument_name_to_instrument_class()
instrumenttools.AltoTrombone (class in ab- (in module ab-
jad.tools.instrumenttools.AltoTrombone.AltoTrombone), jad.tools.instrumenttools.default_instrument_name_to_instrument
504 769

instrumenttools.BaritoneSaxophone (class in ab- instrumenttools.EFlatClarinet (class in ab-
jad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone instrumenttools.EFlatClarinet.EFlatClarinet),
516 612

instrumenttools.BaritoneVoice (class in ab- instrumenttools.EnglishHorn (class in ab-
jad.tools.instrumenttools.BaritoneVoice.BaritoneVoice), jad.tools.instrumenttools.EnglishHorn.EnglishHorn),
522 618

instrumenttools.BassClarinet (class in ab- instrumenttools.Flute (class in ab-
jad.tools.instrumenttools.BassClarinet.BassClarinet), jad.tools.instrumenttools.Flute.Flute), 624
528 instrumenttools.FrenchHorn (class in ab-
instrumenttools.BassFlute (class in ab- jad.tools.instrumenttools.FrenchHorn.FrenchHorn),
jad.tools.instrumenttools.BassFlute.BassFlute), 629

instrumenttools.Glockenspiel (class in abjad.tools.instrumenttools.Glockenspiel.Glockenspiel), 635	jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone (class in abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone), 692
instrumenttools.Guitar (class in abjad.tools.instrumenttools.Guitar.Guitar), 641	instrumenttools.SopranoSaxophone (class in abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone), 698
instrumenttools.Harp (class in abjad.tools.instrumenttools.Harp.Harp), 647	instrumenttools.SopranoVoice (class in abjad.tools.instrumenttools.SopranoVoice.SopranoVoice), 704
instrumenttools.Harpsichord (class in abjad.tools.instrumenttools.Harpsichord.Harpsichord), 653	instrumenttools.TenorSaxophone (class in abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone), 710
instrumenttools.InstrumentInventory (class in abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory), 658	instrumenttools.TenorTrombone (class in abjad.tools.instrumenttools.TenorTrombone.TenorTrombone), 716
instrumenttools.iterate_notes_and_chords_in_expr_outside_traditional_instrument_ranges() (in module abjad.tools.instrumenttools.iterate_notes_and_chords_in_expr_outside_traditional_instrument_ranges), 769	instrumenttools.TenorVoice (class in abjad.tools.instrumenttools.TenorVoice.TenorVoice), 769
instrumenttools.list_instrument_names() (in module abjad.tools.instrumenttools.list_instrument_names), 770	instrumenttools.transpose_from_fingered_pitch_to_sounding_pitch() (in module abjad.tools.instrumenttools.transpose_from_fingered_pitch_to_sounding_pitch), 773
instrumenttools.list_instruments() (in module abjad.tools.instrumenttools.list_instruments), 771	instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch() (in module abjad.tools.instrumenttools.transpose_from_sounding_pitch_to_fingered_pitch), 774
instrumenttools.list_primary_instruments() (in module abjad.tools.instrumenttools.list_primary_instruments), 771	instrumenttools.Trumpet (class in abjad.tools.instrumenttools.Trumpet.Trumpet), 728
instrumenttools.list_secondary_instruments() (in module abjad.tools.instrumenttools.list_secondary_instruments), 771	instrumenttools.Tuba (class in abjad.tools.instrumenttools.Tuba.Tuba), 734
instrumenttools.Marimba (class in abjad.tools.instrumenttools.Marimba.Marimba), 662	instrumenttools.UntunedPercussion (class in abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion), 740
instrumenttools.MezzoSopranoVoice (class in abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice), 668	instrumenttools.Vibraphone (class in abjad.tools.instrumenttools.Vibraphone.Vibraphone), 746
instrumenttools.notes_and_chords_in_expr_are_on_expected_instruments() (in module abjad.tools.instrumenttools.notes_and_chords_in_expr_are_on_expected_instruments), 772	instrumenttools.Viola (class in abjad.tools.instrumenttools.Viola.Viola), 752
instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges() (in module abjad.tools.instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges), 773	instrumenttools.Violin (class in abjad.tools.instrumenttools.Violin.Violin), 758
instrumenttools.Oboe (class in abjad.tools.instrumenttools.Oboe.Oboe), 674	instrumenttools.Xylophone (class in abjad.tools.instrumenttools.Xylophone.Xylophone), 764
instrumenttools.Piano (class in abjad.tools.instrumenttools.Piano.Piano), 680	intersection() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet), 1150
instrumenttools.Piccolo (class in abjad.tools.instrumenttools.Piccolo.Piccolo), 686	intersection() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet), 1155
instrumenttools.SopraninoSaxophone (class in abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone), 692	intersection() (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet), 1089
	intersection() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet), 1097
	intersection() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet), 1150

method), 1162

intersection() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1187

intersection() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1200

intersection() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1212

intersection() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1219

intersection() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1240

intersection() (abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet attribute), 1178

intersection() (abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet method), 1111

intersection() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet method), 1120

intersection() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet method), 1127

intersection() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass attribute), 1807

intersection() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1804

interval_class (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.ChromaticIntervalObject attribute), 1066

interval_class (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.CounterpointIntervalObject attribute), 1072

interval_class (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.DiatonicIntervalObject attribute), 1076

interval_class (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval attribute), 1131

interval_class (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval attribute), 1142

interval_class (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval attribute), 1146

interval_class (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.HarmonicIntervalObject attribute), 1084

interval_class (abjad.tools.pitchtools.IntervalObject.IntervalObject.IntervalObject attribute), 1092

interval_class (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval attribute), 1175

interval_class (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval attribute), 1189

interval_class (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval attribute), 1193

interval_class (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject.MelodicIntervalObject attribute), 1102

interval_class_numbers (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment.IntervalClassObjectSegment attribute), 1087

interval_class_numbers (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment attribute), 1159

interval_class_numbers (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment attribute), 1178

interval_classes (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment attribute), 1137

interval_classes (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment attribute), 1142

interval_classes (abjad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment.IntervalClassObjectSegment attribute), 1087

interval_classes (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment.IntervalObjectSegment attribute), 1092

interval_classes (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment attribute), 1159

interval_classes (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment attribute), 1164

interval_classes (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment attribute), 1178

interval_classes (abjad.tools.pitchtools.MelodicDiatonicIntervalClassSegment.MelodicDiatonicIntervalClassSegment attribute), 1187

interval_classes (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator attribute), 1807

interval_of_transposition (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject.ChromaticIntervalObject attribute), 487

interval_of_transposition (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject.CounterpointIntervalObject attribute), 1072

interval_of_transposition (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.DiatonicIntervalObject attribute), 1076

interval_of_transposition (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval attribute), 499

interval_of_transposition (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval attribute), 1142

interval_of_transposition (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval attribute), 1146

interval_of_transposition (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.HarmonicIntervalObject attribute), 517

interval_of_transposition (abjad.tools.pitchtools.IntervalObject.IntervalObject.IntervalObject attribute), 1092

interval_of_transposition (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval attribute), 1175

interval_of_transposition (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval attribute), 529

interval_of_transposition (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval attribute), 1193

interval_of_transposition (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject.MelodicIntervalObject attribute), 1102

interval_of_transposition (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 559

interval_of_transposition (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 611

interval_of_transposition (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 617

interval_of_transposition (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 553

interval_of_transposition jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet. attribute), 511	(ab- interval_of_transposition jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice. attribute), 669	(ab- interval_of_transposition jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice. attribute), 669
interval_of_transposition jad.tools.instrumenttools.Cello.Cello.Cello attribute), 565	(ab- interval_of_transposition jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 675	(ab- interval_of_transposition jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 675
interval_of_transposition jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA. attribute), 571	(ab- interval_of_transposition jad.tools.instrumenttools.Piano.Piano.Piano attribute), 681	(ab- interval_of_transposition jad.tools.instrumenttools.Piano.Piano.Piano attribute), 681
interval_of_transposition jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass. attribute), 577	(ab- interval_of_transposition jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 687	(ab- interval_of_transposition jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 687
interval_of_transposition jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet. attribute), 583	(ab- interval_of_transposition jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone. attribute), 693	(ab- interval_of_transposition jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone. attribute), 693
interval_of_transposition jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute. attribute), 589	(ab- interval_of_transposition jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone. attribute), 699	(ab- interval_of_transposition jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone. attribute), 699
interval_of_transposition jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon. attribute), 601	(ab- interval_of_transposition jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice. attribute), 705	(ab- interval_of_transposition jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice. attribute), 705
interval_of_transposition jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone. attribute), 595	(ab- interval_of_transposition jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone. attribute), 711	(ab- interval_of_transposition jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone. attribute), 711
interval_of_transposition jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice. attribute), 607	(ab- interval_of_transposition jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone. attribute), 717	(ab- interval_of_transposition jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone. attribute), 717
interval_of_transposition jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet. attribute), 613	(ab- interval_of_transposition jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice. attribute), 723	(ab- interval_of_transposition jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice. attribute), 723
interval_of_transposition jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn. attribute), 619	(ab- interval_of_transposition jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 729	(ab- interval_of_transposition jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 729
interval_of_transposition jad.tools.instrumenttools.Flute.Flute.Flute attribute), 625	(ab- interval_of_transposition jad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 735	(ab- interval_of_transposition jad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 735
interval_of_transposition jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn. attribute), 630	(ab- interval_of_transposition jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion. attribute), 741	(ab- interval_of_transposition jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion. attribute), 741
interval_of_transposition jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel. attribute), 636	(ab- interval_of_transposition jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 747	(ab- interval_of_transposition jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 747
interval_of_transposition jad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 642	(ab- interval_of_transposition jad.tools.instrumenttools.Viola.Viola.Viola attribute), 753	(ab- interval_of_transposition jad.tools.instrumenttools.Viola.Viola.Viola attribute), 753
interval_of_transposition jad.tools.instrumenttools.Harp.Harp.Harp attribute), 648	(ab- interval_of_transposition jad.tools.instrumenttools.Violin.Violin.Violin attribute), 759	(ab- interval_of_transposition jad.tools.instrumenttools.Violin.Violin.Violin attribute), 759
interval_of_transposition jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord. attribute), 654	(ab- interval_of_transposition jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 765	(ab- interval_of_transposition jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 765
interval_of_transposition jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 663	(ab- interval_string (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject. attribute), 1076	(ab- interval_string (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject. attribute), 1076
	interval_string (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval. attribute), 1076	interval_string (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval. attribute), 1076

attribute), 1146	attribute), 1215
interval_string (abjad.tools.pitchtools.MelodicDiatonicIntervalClassSegment), 1193	interval_string (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet), 1215
intervals (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment), 1137	intervals (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment), 1137
intervals (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment), 1152	intervals (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment), 1152
intervals (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment), 1095	intervals (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment), 1095
intervals (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment), 1169	intervals (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment), 1169
intervals (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment), 1183	intervals (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment), 1183
intervals (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment), 1197	intervals (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment), 1197
intervals (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin), 1701	intervals (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin), 1701
intervals (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree), 1712	intervals (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree), 1712
intervals (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary), 1725	intervals (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary), 1725
intervals (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator), 1807	intervals (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator), 1807
introspectiontools.get_current_function_name() (in module abjad.tools.introspectiontools.get_current_function_name), 2034	introspectiontools.get_current_function_name() (in module abjad.tools.introspectiontools.get_current_function_name), 2034
introspectiontools.klass_to_tools_package_qualified_class_name() (in module abjad.tools.introspectiontools.klass_to_tools_package_qualified_class_name), 2034	introspectiontools.klass_to_tools_package_qualified_class_name() (in module abjad.tools.introspectiontools.klass_to_tools_package_qualified_class_name), 2034
inversion (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass), 1803	inversion (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass), 1803
inversion (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator), 1807	inversion (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator), 1807
inversion (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction), 1822	inversion (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction), 1822
inversion_equivalent_chromatic_interval_class (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval), 1193	inversion_equivalent_chromatic_interval_class (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval), 1193
inversion_equivalent_chromatic_interval_class_numbers (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet), 1161	inversion_equivalent_chromatic_interval_class_numbers (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet), 1161
inversion_equivalent_chromatic_interval_class_segment (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment), 1215	inversion_equivalent_chromatic_interval_class_segment (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment), 1215
inversion_equivalent_chromatic_interval_class_segment (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment), 1236	inversion_equivalent_chromatic_interval_class_segment (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment), 1236
inversion_equivalent_chromatic_interval_class_segment (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow), 1266	inversion_equivalent_chromatic_interval_class_segment (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow), 1266
inversion_equivalent_chromatic_interval_class_set (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment), 1215	inversion_equivalent_chromatic_interval_class_set (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment), 1215

iotools.redo() (in module `abjad.tools.iotools.redo`), 778
iotools.save_last_ly_as() (in module `abjad.tools.iotools.save_last_ly_as`), 778
iotools.save_last_pdf_as() (in module `abjad.tools.iotools.save_last_pdf_as`), 778
iotools.show() (in module `abjad.tools.iotools.show`), 779
iotools.spawn_subprocess() (in module `abjad.tools.iotools.spawn_subprocess`), 779
iotools.write_expr_to_ly() (in module `abjad.tools.iotools.write_expr_to_ly`), 779
iotools.write_expr_to_pdf() (in module `abjad.tools.iotools.write_expr_to_pdf`), 779
iotools.z() (in module `abjad.tools.iotools.z`), 780
is_abstract (`abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter` attribute), 1959
is_adjusted (`abjad.tools.pitchtools.Accidental.Accidental` attribute), 1129
is_at_level() (`abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree` method), 1428
is_at_level() (`abjad.tools.sequencetools.Tree.Tree.Tree` method), 1440
is_augmentation (`abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet` attribute), 1829
is_augmentation (`abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet` attribute), 1840
is_binary (`abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure` attribute), 978
is_binary (`abjad.tools.measuretools.DynamicMeasure.DynamicMeasure` attribute), 990
is_binary (`abjad.tools.measuretools.Measure.Measure.Measure` attribute), 1001
is_binary (`abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet` attribute), 1830
is_binary (`abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet` attribute), 1840
is_cautionary (`abjad.tools.notetools.NoteHead.NoteHead.NoteHead` attribute), 1042
is_closed (`abjad.tools.mathtools.BoundedObject.BoundedObject` attribute), 945
is_congruent_base() (`abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression` method), 1493
is_contained_by_interval() (`abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval` method), 1708
is_contained_by_interval() (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin` method), 1702
is_contained_by_interval() (`abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin` method), 1705
is_contained_by_interval() (`abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree` method), 1717
is_contained_by_interval() (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary` method), 1736
is_container_of_interval() (`abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval` method), 1708
is_container_of_interval() (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin` method), 1702
is_container_of_interval() (`abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin` method), 1705
is_container_of_interval() (`abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree` method), 1717
is_container_of_interval() (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary` method), 1736
is_cyclic (`abjad.tools.datastructuretools.Digraph.Digraph.Digraph` attribute), 1900
is_defective (`abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn` attribute), 1055
is_defective (`abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow` attribute), 1057
is_diminution (`abjad.tools.timetokentools.EqualDivisionTimeTokenMaker.EqualDivisionTimeTokenMaker` attribute), 1769
is_diminution (`abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet` attribute), 1830
is_diminution (`abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet` attribute), 1840
is_doubling (`abjad.tools.scoretools.Performer.Performer.Performer` attribute), 1130
is_empty (`abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator` attribute), 1820
is_equivalent_under_transposition() (`abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment` method), 1210
is_equivalent_under_transposition() (`abjad.tools.tonalitytools.Scale.Scale.Scale` method), 1817
is_forced (`abjad.tools.notetools.NoteHead.NoteHead.NoteHead` attribute), 1042
is_formatted_when_empty (`abjad.tools.lilypondfiletools.AttributedBlock.AttributedBlock` attribute), 1558
is_formatted_when_empty (`abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock` attribute), 1561
is_formatted_when_empty (`abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock` attribute), 1564
is_formatted_when_empty (`abjad.tools.lilypondfiletools.AttributedBlock.AttributedBlock` attribute), 1558

attribute), 337

is_parallel (abjad.tools.containertools.Container.Container.ContainerContainer attribute), 344

is_parallel (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer attribute), 351

is_parallel (abjad.tools.contexttools.Context.Context.Context attribute), 387

is_parallel (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer attribute), 477

is_parallel (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 983

is_parallel (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 994

is_parallel (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1005

is_parallel (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1372

is_parallel (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1388

is_parallel (abjad.tools.scoretools.Score.Score.Score attribute), 1397

is_parallel (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1406

is_parallel (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1657

is_parallel (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1666

is_parallel (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1833

is_parallel (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1843

is_parallel (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1868

is_pitch_class_unique (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet attribute), 1218

is_pitched (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1342

is_pitched (abjad.tools.tietools.TieChain.TieChain.TieChain attribute), 1682

is_primary_instrument (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 487

is_primary_instrument (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 493

is_primary_instrument (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone attribute), 499

is_primary_instrument (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone attribute), 505

is_primary_instrument (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone attribute), 517

is_primary_instrument (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 553

is_primary_instrument (abjad.tools.instrumenttools.BassTrombone.BassTrombone attribute), 547

is_primary_instrument (abjad.tools.instrumenttools.BassClarinet.BassClarinet attribute), 529

is_primary_instrument (abjad.tools.instrumenttools.BassFlute.BassFlute attribute), 559

is_primary_instrument (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone attribute), 541

is_primary_instrument (abjad.tools.instrumenttools.Bassoon.Bassoon attribute), 559

is_primary_instrument (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet attribute), 511

is_primary_instrument (abjad.tools.instrumenttools.Cello.Cello.Cello attribute), 571

is_primary_instrument (abjad.tools.instrumenttools.Contrabass.Contrabass attribute), 577

is_primary_instrument (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet attribute), 581

is_primary_instrument (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute attribute), 589

is_primary_instrument (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon attribute), 601

is_primary_instrument (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone attribute), 595

is_primary_instrument (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice attribute), 607

is_primary_instrument (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet attribute), 613

is_primary_instrument (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn attribute), 613

attribute), 619	attribute), 729
is_primary_instrument jad.tools.instrumenttools.Flute.Flute.Flute attribute), 625	(ab- is_primary_instrument jad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 735
is_primary_instrument jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 630	(ab- is_primary_instrument jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion. attribute), 741
is_primary_instrument jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 636	(ab- is_primary_instrument jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 747
is_primary_instrument jad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 642	(ab- is_primary_instrument jad.tools.instrumenttools.Viola.Viola.Viola attribute), 753
is_primary_instrument jad.tools.instrumenttools.Harp.Harp.Harp attribute), 648	(ab- is_primary_instrument jad.tools.instrumenttools.Violin.Violin.Violin attribute), 759
is_primary_instrument jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord attribute), 654	(ab- is_primary_instrument jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 765
is_primary_instrument jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 663	(ab- is_rectangular (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1050
is_primary_instrument jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice attribute), 669	(ab- is_right_closed (abjad.tools.mathtools.BoundedObject.BoundedObject.Bou attribute), 945
is_primary_instrument jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 675	(ab- is_secondary_instrument jad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 487
is_primary_instrument jad.tools.instrumenttools.Piano.Piano.Piano attribute), 681	(ab- is_secondary_instrument jad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 493
is_primary_instrument jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 687	(ab- is_secondary_instrument jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSax attribute), 499
is_primary_instrument jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone attribute), 693	(ab- is_secondary_instrument jad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrom attribute), 505
is_primary_instrument jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 699	(ab- is_secondary_instrument jad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone attribute), 517
is_primary_instrument jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 705	(ab- is_secondary_instrument jad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneV attribute), 523
is_primary_instrument jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 711	(ab- is_secondary_instrument jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 529
is_primary_instrument jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 717	(ab- is_secondary_instrument jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 535
is_primary_instrument jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 723	(ab- is_secondary_instrument jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 559
is_primary_instrument jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet	(ab- is_secondary_instrument jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSa

attribute), 541	attribute), 648
is_secondary_instrument jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone	is_secondary_instrument (ab- jad.tools.instrumenttools.Harp. attribute), 547
is_secondary_instrument jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice	is_secondary_instrument (ab- jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 553
is_secondary_instrument jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet	is_secondary_instrument (ab- jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice attribute), 511
is_secondary_instrument jad.tools.instrumenttools.Cello.Cello.Cello	is_secondary_instrument (ab- jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 565
is_secondary_instrument jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA	is_secondary_instrument (ab- jad.tools.instrumenttools.Piano.Piano.Piano attribute), 571
is_secondary_instrument jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass	is_secondary_instrument (ab- jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 577
is_secondary_instrument jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet	is_secondary_instrument (ab- jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone attribute), 583
is_secondary_instrument jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute	is_secondary_instrument (ab- jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone attribute), 589
is_secondary_instrument jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon	is_secondary_instrument (ab- jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 601
is_secondary_instrument jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone	is_secondary_instrument (ab- jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 595
is_secondary_instrument jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice	is_secondary_instrument (ab- jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 607
is_secondary_instrument jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet	is_secondary_instrument (ab- jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 613
is_secondary_instrument jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn	is_secondary_instrument (ab- jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 619
is_secondary_instrument jad.tools.instrumenttools.Flute.Flute.Flute	is_secondary_instrument (ab- jad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 625
is_secondary_instrument jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn	is_secondary_instrument (ab- jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion. attribute), 630
is_secondary_instrument jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel	is_secondary_instrument (ab- jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 636
is_secondary_instrument jad.tools.instrumenttools.Guitar.Guitar.Guitar	is_secondary_instrument (ab- jad.tools.instrumenttools.Viola.Viola.Viola attribute), 642
is_secondary_instrument jad.tools.instrumenttools.Harp.Harp.Harp	is_secondary_instrument (ab- jad.tools.instrumenttools.Violin.Violin.Violin attribute), 648

attribute), 759

is_secondary_instrument (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 765

is_semantic (abjad.tools.contexttools.Context.Context.Context attribute), 385

is_semantic (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1370

is_semantic (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1386

is_semantic (abjad.tools.scoretools.Score.Score.Score attribute), 1395

is_semantic (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1404

is_semantic (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1655

is_semantic (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1664

is_semantic (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1866

is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1708

is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeIntervalAggregate.TimeIntervalAggregate.TimeIntervalAggregate method), 1702

is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.TimeIntervalMixin method), 1705

is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method), 1717

is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1737

is_tempo_mark_token() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method), 413

is_tertian (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment attribute), 1169

is_transposed_subset() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1240

is_transposed_superset() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1241

is_transposing (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 487

is_transposing (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 493

is_transposing (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 499

is_transposing (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 505

is_transposing (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone attribute), 517

is_transposing (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice attribute), 523

is_transposing (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 529

is_transposing (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 535

is_transposing (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 559

is_transposing (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 541

is_transposing (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 547

is_transposing (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 553

is_transposing (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 511

is_transposing (abjad.tools.instrumenttools.Cello.Cello.Cello attribute), 565

is_transposing (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute), 571

is_transposing (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass attribute), 577

is_transposing (abjad.tools.instrumenttools.ContrabassCello.ContrabassCello.ContrabassCello attribute), 583

is_transposing (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute attribute), 589

is_transposing (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon attribute), 601

is_transposing (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone attribute), 595

is_transposing (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice attribute), 607

is_transposing (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute), 613

is_transposing (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 619

is_transposing (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 636

is_transposing (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 642

is_transposing (abjad.tools.instrumenttools.Harp.Harp.Harp attribute), 648

is_transposing (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord attribute), 654

is_transposing (abjad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 663

is_transposing (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice attribute), 669

issuperset()	(abjad.tools.pitchtools.HarmonicChromaticIntervalClassSet.HarmonicChromaticIntervalClassSet.HarmonicChromaticIntervalClassSet)	method), 1140
issuperset()	(abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet)	method), 1150
issuperset()	(abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet)	method), 1155
issuperset()	(abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet.IntervalClassObjectSet)	method), 1090
issuperset()	(abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet)	method), 1098
issuperset()	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet)	method), 1162
issuperset()	(abjad.tools.pitchtools.MelodicChromaticIntervalClassSet.MelodicChromaticIntervalClassSet.MelodicChromaticIntervalClassSet)	method), 1187
issuperset()	(abjad.tools.pitchtools.MelodicDiatonicIntervalClassSet.MelodicDiatonicIntervalClassSet.MelodicDiatonicIntervalClassSet)	method), 1200
issuperset()	(abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet)	method), 1213
issuperset()	(abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet)	method), 1219
issuperset()	(abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet)	method), 1241
issuperset()	(abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet)	method), 1112
issuperset()	(abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet)	method), 1121
issuperset()	(abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet)	method), 1127
issuperset()	(abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass)	method), 1805
items()	(abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig)	method), 1896
items()	(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary)	method), 1903
items()	(abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph)	method), 1966
items()	(abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector)	method), 1135
items()	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector)	method), 1165
items()	(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector)	method), 1172
items()	(abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector)	method), 1181
items()	(abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector)	method), 1222
items()	(abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector)	method), 1244
items()	(abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector)	method), 1114
items()	(abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval)	method), 1708
items()	(abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary)	method), 1737

method), 1737

K

key() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy
method), 2035

key_signature (abjad.tools.tonalitytools.Scale.Scale.Scale
attribute), 1816

keys() (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig
method), 1896

keys() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary
method), 1903

keys() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph
method), 1966

keys() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector
method), 1135

keys() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector
method), 1165

keys() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector
method), 1172

keys() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector
method), 1181

keys() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector
method), 1222

keys() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector
method), 1244

keys() (abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector
method), 1114

keys() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval
method), 1708

keys() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary
method), 1737

kill() (abjad.tools.documentationtools.Pipe.Pipe.Pipe
method), 1970

kind (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer
attribute), 477

kind (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner
attribute), 1599

L

labeltools.color_chord_note_heads_in_expr_by_pitch_class_color_map()
(in module abjad.tools.labeltools.color_chord_note_heads_in_expr_by_pitch_class_color_map), 800

labeltools.color_contents_of_container() (in module abjad.tools.labeltools.color_contents_of_container), 801

labeltools.color_leaf() (in module abjad.tools.labeltools.color_leaf), 801

labeltools.color_leaves_in_expr() (in module abjad.tools.labeltools.color_leaves_in_expr), 802

labeltools.color_measure() (in module abjad.tools.labeltools.color_measure), 803

labeltools.color_nonbinary_measures_in_expr()
(in module abjad.tools.labeltools.color_nonbinary_measures_in_expr), 803

labeltools.color_note_head_by_numbered_chromatic_pitch_class_color_map()
(in module abjad.tools.labeltools.color_note_head_by_numbered_chromatic_pitch_class_color_map), 804

labeltools.label_leaves_in_expr_with_inversion_equivalent_chromatic_intervals()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_inversion_equivalent_chromatic_intervals), 804

labeltools.label_leaves_in_expr_with_leaf_depth()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_leaf_depth), 805

labeltools.label_leaves_in_expr_with_leaf_durations()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_leaf_durations), 806

labeltools.label_leaves_in_expr_with_leaf_indices()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_leaf_indices), 806

labeltools.label_leaves_in_expr_with_leaf_numbers()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_leaf_numbers), 807

labeltools.label_leaves_in_expr_with_melodic_chromatic_interval_classes()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_melodic_chromatic_interval_classes), 807

labeltools.label_leaves_in_expr_with_melodic_chromatic_intervals()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_melodic_chromatic_intervals), 808

labeltools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes), 809

labeltools.label_leaves_in_expr_with_melodic_counterpoint_intervals()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_melodic_counterpoint_intervals), 809

labeltools.label_leaves_in_expr_with_melodic_diatonic_interval_classes()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_melodic_diatonic_interval_classes), 809

labeltools.label_leaves_in_expr_with_melodic_diatonic_intervals()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_melodic_diatonic_intervals), 810

labeltools.label_leaves_in_expr_with_pitch_class_numbers()
(in module abjad.tools.labeltools.label_leaves_in_expr_with_pitch_class_numbers), 810

jad.tools.labeltools.label_leaves_in_expr_with_pitch_numbers() (in module abjad.tools.labeltools, 810)
 labeltools.label_leaves_in_expr_with_pitch_numbers() (in module abjad.tools.labeltools, 820)
 jad.tools.labeltools.label_leaves_in_expr_with_pitch_numbers() (in module abjad.tools.labeltools, 811)
 labeltools.label_leaves_in_expr_with_prolated_leaf_duration() (in module abjad.tools.labeltools, 821)
 jad.tools.labeltools.label_leaves_in_expr_with_prolated_leaf_duration() (in module abjad.tools.labeltools, 811)
 labeltools.label_leaves_in_expr_with_tuplet_depth() (in module abjad.tools.labeltools, 823)
 jad.tools.labeltools.label_leaves_in_expr_with_tuplet_depth() (in module abjad.tools.labeltools, 811)
 labeltools.label_leaves_in_expr_with_written_leaf_duration() (in module abjad.tools.labeltools, 1701)
 jad.tools.labeltools.label_leaves_in_expr_with_written_leaf_duration() (in module abjad.tools.labeltools, 812)
 labeltools.label_notes_in_expr_with_note_indices() (in module abjad.tools.labeltools, 1725)
 jad.tools.labeltools.label_notes_in_expr_with_note_indices() (in module abjad.tools.labeltools, 812)
 labeltools.label_tie_chains_in_expr_with_prolated_tie_chain_duration() (in module abjad.tools.labeltools, 1712)
 jad.tools.labeltools.label_tie_chains_in_expr_with_prolated_tie_chain_duration() (in module abjad.tools.labeltools, 812)
 labeltools.label_tie_chains_in_expr_with_tie_chain_duration() (in module abjad.tools.labeltools, 825)
 jad.tools.labeltools.label_tie_chains_in_expr_with_tie_chain_duration() (in module abjad.tools.labeltools, 813)
 labeltools.label_tie_chains_in_expr_with_written_tie_chain_duration() (in module abjad.tools.labeltools, 826)
 jad.tools.labeltools.label_tie_chains_in_expr_with_written_tie_chain_duration() (in module abjad.tools.labeltools, 814)
 labeltools.label_vertical_moments_in_expr_with_chromatic_interval_classes() (in module abjad.tools.labeltools, 827)
 jad.tools.labeltools.label_vertical_moments_in_expr_with_chromatic_interval_classes() (in module abjad.tools.labeltools, 814)
 labeltools.label_vertical_moments_in_expr_with_chromatic_intervals() (in module abjad.tools.labeltools, 823)
 jad.tools.labeltools.label_vertical_moments_in_expr_with_chromatic_intervals() (in module abjad.tools.labeltools, 815)
 labeltools.label_vertical_moments_in_expr_with_counterpoint_intervals() (in module abjad.tools.labeltools, 1769)
 jad.tools.labeltools.label_vertical_moments_in_expr_with_counterpoint_intervals() (in module abjad.tools.labeltools, 817)
 labeltools.label_vertical_moments_in_expr_with_diatonic_intervals() (in module abjad.tools.labeltools, 828)
 jad.tools.labeltools.label_vertical_moments_in_expr_with_diatonic_intervals() (in module abjad.tools.labeltools, 818)
 labeltools.label_vertical_moments_in_expr_with_interval_classes() (in module abjad.tools.labeltools, 1038)
 jad.tools.labeltools.label_vertical_moments_in_expr_with_interval_classes() (in module abjad.tools.labeltools, 819)

[leaf_index \(abjad.tools.resttools.Rest.Rest.Rest attribute\), 1311](#)
[leaf_index \(abjad.tools.skiptools.Skip.Skip.Skip attribute\), 1495](#)
[leaftools.all_are_leaves\(\) \(in module abjad.tools.leaftools.all_are_leaves\), 831](#)
[leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_durations\(\) \(in module abjad.tools.leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_durations\), 831](#)
[leaftools.copy_written_duration_and_multiplier_from_leaf_to_leaf\(\) \(in module abjad.tools.leaftools.copy_written_duration_and_multiplier_from_leaf_to_leaf\), 832](#)
[leaftools.divide_leaf_meiotically\(\) \(in module abjad.tools.leaftools.divide_leaf_meiotically\), 832](#)
[leaftools.divide_leaves_in_expr_meiotically\(\) \(in module abjad.tools.leaftools.divide_leaves_in_expr_meiotically\), 832](#)
[leaftools.expr_has_leaf_with_dotted_written_duration\(\) \(in module abjad.tools.leaftools.expr_has_leaf_with_dotted_written_duration\), 833](#)
[leaftools.fuse_leaves\(\) \(in module abjad.tools.leaftools.fuse_leaves\), 834](#)
[leaftools.fuse_leaves_in_container_once_by_counts\(\) \(in module abjad.tools.leaftools.fuse_leaves_in_container_once_by_counts\), 834](#)
[leaftools.fuse_leaves_in_tie_chain_by_immediate_parent\(\) \(in module abjad.tools.leaftools.fuse_leaves_in_tie_chain_by_immediate_parent\), 834](#)
[leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_and_without_change\(\) \(in module abjad.tools.leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_and_without_change\), 835](#)
[leaftools.get_composite_offset_difference_series_from_leaves_in_expr\(\) \(in module abjad.tools.leaftools.get_composite_offset_difference_series_from_leaves_in_expr\), 835](#)
[leaftools.get_composite_offset_series_from_leaves_in_expr\(\) \(in module abjad.tools.leaftools.get_composite_offset_series_from_leaves_in_expr\), 836](#)
[leaftools.get_leaf_at_index_in_measure_number_in_expr\(\) \(in module abjad.tools.leaftools.get_leaf_at_index_in_measure_number_in_expr\), 837](#)
[leaftools.get_leaf_in_expr_with_maximum_prolated_duration\(\) \(in module abjad.tools.leaftools.get_leaf_in_expr_with_maximum_prolated_duration\), 837](#)
[leaftools.get_leaf_in_expr_with_minimum_prolated_duration\(\) \(in module abjad.tools.leaftools.get_leaf_in_expr_with_minimum_prolated_duration\), 838](#)
[leaftools.get_nth_leaf_in_expr\(\) \(in module abjad.tools.leaftools.get_nth_leaf_in_expr\), 838](#)
[leaftools.get_nth_prolated_leaf_in_expr\(\) \(in module abjad.tools.leaftools.get_nth_prolated_leaf_in_expr\), 839](#)
[leaftools.is_bar_line_crossing_leaf\(\) \(in module abjad.tools.leaftools.is_bar_line_crossing_leaf\), 840](#)
[leaftools.Leaf \(class in abjad.tools.leaftools.Leaf.Leaf\), 828](#)
[leaftools.list_prolated_durations_of_leaves_in_expr\(\) \(in module abjad.tools.leaftools.list_prolated_durations_of_leaves_in_expr\), 840](#)
[leaftools.list_written_durations_of_leaves_in_expr\(\) \(in module abjad.tools.leaftools.list_written_durations_of_leaves_in_expr\), 840](#)
[leaftools.make_leaves\(\) \(in module abjad.tools.leaftools.make_leaves\), 841](#)
[leaftools.make_leaves_from_note_value_signal\(\) \(in module abjad.tools.leaftools.make_leaves_from_note_value_signal\), 842](#)
[leaftools.make_tied_leaf\(\) \(in module abjad.tools.leaftools.make_tied_leaf\), 842](#)
[leaftools.move_initial_rests_from_sequence\(\) \(in module abjad.tools.leaftools.move_initial_rests_from_sequence\), 843](#)
[leaftools.remove_leaf_and_shrink_durated_parent_containers\(\) \(in module abjad.tools.leaftools.remove_leaf_and_shrink_durated_parent_containers\), 843](#)
[leaftools.remove_outer_rests_from_sequence\(\) \(in module abjad.tools.leaftools.remove_outer_rests_from_sequence\), 844](#)
[leaftools.remove_terminal_rests_from_sequence\(\) \(in module abjad.tools.leaftools.remove_terminal_rests_from_sequence\), 845](#)
[leaftools.repeat_leaf\(\) \(in module abjad.tools.leaftools.repeat_leaf\), 845](#)
[leaftools.repeat_leaves_in_expr\(\) \(in module abjad.tools.leaftools.repeat_leaves_in_expr\), 845](#)

<code>leaftools.replace_leaves_in_expr_with_parallel_voices()</code>	<code>(in module abjad.tools.leaftools.replace_leaves_in_expr_with_parallel_voices)</code> , 846	<code>leaves (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1370</code>
<code>leaftools.replace_leaves_in_expr_with_parallel_voices()</code>	<code>(in module abjad.tools.leaftools.replace_leaves_in_expr_with_parallel_voices)</code> , 846	<code>leaves (abjad.tools.scoretools.Score.Score.Score attribute), 1386</code>
<code>leaftools.replace_leaves_in_expr_with_parallel_voices()</code>	<code>(in module abjad.tools.leaftools.replace_leaves_in_expr_with_parallel_voices)</code> , 848	<code>leaves (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1395</code>
<code>leaftools.rest_leaf_at_offset()</code>	<code>(in module abjad.tools.leaftools.rest_leaf_at_offset)</code> , 850	<code>leaves (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner attribute), 1404</code>
<code>leaftools.scale_preprolated_leaf_duration()</code>	<code>(in module abjad.tools.leaftools.scale_preprolated_leaf_duration)</code> , 850	<code>leaves (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner attribute), 1513</code>
<code>leaftools.set_preprolated_leaf_duration()</code>	<code>(in module abjad.tools.leaftools.set_preprolated_leaf_duration)</code> , 851	<code>leaves (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1520</code>
<code>leaftools.show_leaves()</code>	<code>(in module abjad.tools.leaftools.show_leaves)</code> , 853	<code>leaves (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1527</code>
<code>leaftools.split_leaf_at_offset()</code>	<code>(in module abjad.tools.leaftools.split_leaf_at_offset)</code> , 853	<code>leaves (abjad.tools.spannertools.DirectedSpanner.DirectedSpanner.DirectedSpanner attribute), 1536</code>
<code>leaftools.split_leaf_at_offsets()</code>	<code>(in module abjad.tools.leaftools.split_leaf_at_offsets)</code> , 855	<code>leaves (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner attribute), 1501</code>
<code>leaftools.yield_groups_of_mixed_notes_and_chords_in_sequence()</code>	<code>(in module abjad.tools.leaftools.yield_groups_of_mixed_notes_and_chords_in_sequence)</code> , 857	<code>leaves (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner attribute), 1543</code>
<code>leaves (abjad.tools.beamttools.BeamSpanner.BeamSpanner.BeamSpanner attribute), 224</code>		<code>leaves (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute), 1550</code>
<code>leaves (abjad.tools.beamttools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner attribute), 231</code>		<code>leaves (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner attribute), 1557</code>
<code>leaves (abjad.tools.beamttools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 239</code>		<code>leaves (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner attribute), 1565</code>
<code>leaves (abjad.tools.beamttools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 248</code>		<code>leaves (abjad.tools.beamttools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner attribute), 1571</code>
<code>leaves (abjad.tools.beamttools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner attribute), 256</code>		<code>leaves (abjad.tools.beamttools.OctavationSpanner.OctavationSpanner.OctavationSpanner attribute), 1578</code>
<code>leaves (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 336</code>		<code>leaves (abjad.tools.beamttools.PiccoloSpanner.PiccoloSpanner.PiccoloSpanner attribute), 1585</code>
<code>leaves (abjad.tools.containertools.Container.Container.Container attribute), 342</code>		<code>leaves (abjad.tools.beamttools.PredefinedComplexBeamSpanner.PredefinedComplexBeamSpanner.PredefinedComplexBeamSpanner attribute), 1592</code>
<code>leaves (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer attribute), 350</code>		<code>leaves (abjad.tools.beamttools.PianoPeakSpanner.PianoPeakSpanner.PianoPeakSpanner attribute), 1598</code>
<code>leaves (abjad.tools.contexttools.Context.Context.Context attribute), 385</code>		<code>leaves (abjad.tools.beamttools.SlurSpanner.SlurSpanner.SlurSpanner attribute), 1605</code>
<code>leaves (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer attribute), 475</code>		<code>leaves (abjad.tools.beamttools.Spanner.Spanner.Spanner attribute), 1607</code>
<code>leaves (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure attribute), 980</code>		<code>leaves (abjad.tools.beamttools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner attribute), 1612</code>
<code>leaves (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 991</code>		<code>leaves (abjad.tools.beamttools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner attribute), 1619</code>
<code>leaves (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1002</code>		<code>leaves (abjad.tools.beamttools.TextSpanner.TextSpanner.TextSpanner attribute), 1625</code>
		<code>leaves (abjad.tools.beamttools.TrillSpanner.TrillSpanner.TrillSpanner attribute), 1631</code>
		<code>leaves (abjad.tools.beamttools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1655</code>
		<code>leaves (abjad.tools.beamttools.Staff.Staff.Staff attribute), 1664</code>
		<code>leaves (abjad.tools.tietools.TieChain.TieChain.TieChain attribute), 1682</code>

leaves (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute), 1685

leaves (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1830

leaves (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1840

leaves (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment attribute), 1859

leaves (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1866

leaves_grouped_by_immediate_parents (abjad.tools.tietools.TieChain.TieChain.TieChain attribute), 1683

level (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1425

level (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1438

lexer (abjad.tools.abctools.Parser.Parser.Parser attribute), 1879

lexer (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser attribute), 2046

lexer (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser attribute), 2069

lexer (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser attribute), 2074

lexer (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser attribute), 1344

lexer_rules_object (abjad.tools.abctools.Parser.Parser.Parser attribute), 1879

lexer_rules_object (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser attribute), 2046

lexer_rules_object (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser attribute), 2069

lexer_rules_object (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser attribute), 2074

lexer_rules_object (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser attribute), 1344

likely_instruments_based_on_performer_name (abjad.tools.scoretools.Performer.Performer.Performer attribute), 1380

lilypond_format (abjad.tools.chordtools.Chord.Chord.Chord attribute), 267

lilypond_format (abjad.tools.componenttools.Component.Component.Component attribute), 276

lilypond_format (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 336

lilypond_format (abjad.tools.containertools.Container.Container.Container attribute), 343

lilypond_format (abjad.tools.containertools.FixedDurationCluster.FixedDurationCluster.FixedDurationCluster attribute), 350

lilypond_format (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark attribute), 378

lilypond_format (abjad.tools.contexttools.Context.Context.Context attribute), 385

lilypond_format (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark attribute), 396

lilypond_format (abjad.tools.contexttools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 399

lilypond_format (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark attribute), 403

lilypond_format (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark attribute), 407

lilypond_format (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark attribute), 411

lilypond_format (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark attribute), 419

lilypond_format (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer attribute), 475

lilypond_format (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 487

lilypond_format (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 493

lilypond_format (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 499

lilypond_format (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 505

lilypond_format (abjad.tools.instrumenttools.BaronSaxophone.BaronSaxophone.BaronSaxophone attribute), 517

lilypond_format (abjad.tools.instrumenttools.BaronVoice.BaronVoice.BaronVoice attribute), 529

lilypond_format (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 531

lilypond_format (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 533

lilypond_format (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 535

lilypond_format (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 537

lilypond_format (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 539

lilypond_format (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 541

lilypond_format (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 543

lilypond_format (abjad.tools.instrumenttools.Cello.Cello.Cello attribute), 545

lilypond_format (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute), 547

lilypond_format (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass attribute), 549

lilypond_format (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet attribute), 551

lilypond_format (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute attribute), 553

lilypond_format (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon attribute), 555

lilypond_format (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone attribute), 557

lilypond_format (abjad.tools.instrumenttools.ContraltoVoice.CornettoVoice.CornettoVoice attribute), 607

lilypond_format (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute), 613

lilypond_format (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 619

lilypond_format (abjad.tools.instrumenttools.Flute.Flute.Flute attribute), 625

lilypond_format (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 631

lilypond_format (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 636

lilypond_format (abjad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 642

lilypond_format (abjad.tools.instrumenttools.Harp.Harp.Harp attribute), 648

lilypond_format (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord attribute), 654

lilypond_format (abjad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 663

lilypond_format (abjad.tools.instrumenttools.MezzoSoprano.MezzoSoprano.MezzoSoprano attribute), 669

lilypond_format (abjad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 675

lilypond_format (abjad.tools.instrumenttools.Piano.Piano.Piano attribute), 681

lilypond_format (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 687

lilypond_format (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone attribute), 693

lilypond_format (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 699

lilypond_format (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 705

lilypond_format (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 711

lilypond_format (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 717

lilypond_format (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 723

lilypond_format (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 729

lilypond_format (abjad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 735

lilypond_format (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion attribute), 741

lilypond_format (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 747

lilypond_format (abjad.tools.instrumenttools.Viola.Viola.Viola attribute), 753

lilypond_format (abjad.tools.instrumenttools.Violin.Violin.Violin attribute), 759

lilypond_format (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 765

lilypond_format (abjad.tools.marktools.CornettoVoice.CornettoVoice.CornettoVoice attribute), 828

lilypond_format (abjad.tools.marktools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute), 862

lilypond_format (abjad.tools.marktools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 858

lilypond_format (abjad.tools.marktools.Flute.Flute.Flute attribute), 863

lilypond_format (abjad.tools.marktools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 866

lilypond_format (abjad.tools.marktools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 869

lilypond_format (abjad.tools.marktools.Guitar.Guitar.Guitar attribute), 870

lilypond_format (abjad.tools.marktools.Harp.Harp.Harp attribute), 872

lilypond_format (abjad.tools.marktools.Harpsichord.Harpsichord.Harpsichord attribute), 874

lilypond_format (abjad.tools.marktools.Marimba.Marimba.Marimba attribute), 875

lilypond_format (abjad.tools.marktools.MezzoSoprano.MezzoSoprano.MezzoSoprano attribute), 878

lilypond_format (abjad.tools.marktools.Oboe.Oboe.Oboe attribute), 880

lilypond_format (abjad.tools.marktools.Piano.Piano.Piano attribute), 881

lilypond_format (abjad.tools.marktools.Piccolo.Piccolo.Piccolo attribute), 859

lilypond_format (abjad.tools.marktools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone attribute), 882

lilypond_format (abjad.tools.marktools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 883

lilypond_format (abjad.tools.marktools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 894

lilypond_format (abjad.tools.marktools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 896

lilypond_format (abjad.tools.marktools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 900

lilypond_format (abjad.tools.marktools.TenorVoice.TenorVoice.TenorVoice attribute), 903

lilypond_format (abjad.tools.marktools.Trumpet.Trumpet.Trumpet attribute), 906

lilypond_format (abjad.tools.marktools.Tuba.Tuba.Tuba attribute), 911

lilypond_format (abjad.tools.marktools.UntunedPercussion.UntunedPercussion.UntunedPercussion attribute), 932

lilypond_format (abjad.tools.marktools.Vibraphone.Vibraphone.Vibraphone attribute), 935

lilypond_format (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 980

lilypond_format (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 991

lilypond_format (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1002

lilypond_format (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic, 1033 858
attribute), 1033

lilypond_format (abjad.tools.notetools.Note.Note.Note lilypondfiletools.BookBlock (class in ab-
attribute), 1038 jad.tools.lilypondfiletools.BookBlock.BookBlock),

lilypond_format (abjad.tools.notetools.NoteHead.NoteHead.NoteHead 863
attribute), 1041 lilypondfiletools.BookpartBlock (class in ab-

lilypond_format (abjad.tools.pitchtools.Accidental.Accidental.Accidental, 1129 866
attribute), 1129 jad.tools.lilypondfiletools.BookpartBlock.BookpartBlock),

lilypond_format (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch ab-
attribute), 1204 jad.tools.lilypondfiletools.ContextBlock.ContextBlock),

lilypond_format (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch
attribute), 1225 lilypondfiletools.DateToken (class in ab-

lilypond_format (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest, 1308 870
attribute), 1308 jad.tools.lilypondfiletools.DateToken.DateToken),

lilypond_format (abjad.tools.resttools.Rest.Rest.Rest at- lilypondfiletools.HeaderBlock (class in ab-
tribute), 1311 jad.tools.lilypondfiletools.HeaderBlock.HeaderBlock),

lilypond_format (abjad.tools.schemetools.Scheme.Scheme.Scheme 872
attribute), 1348 lilypondfiletools.LayoutBlock (class in ab-

lilypond_format (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList, 1349 873
attribute), 1349 jad.tools.lilypondfiletools.LayoutBlock.LayoutBlock),

lilypond_format (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor, 1351 874
attribute), 1351 jad.tools.lilypondfiletools.LilyPondFile.LilyPondFile),

lilypond_format (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment
attribute), 1352 lilypondfiletools.LilyPondLanguageToken (class in ab-

lilypond_format (abjad.tools.schemetools.SchemePair.SchemePair.SchemePair, 1354 878
attribute), 1354 jad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken),

lilypond_format (abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector, 1355 879
attribute), 1355 jad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken),

lilypond_format (abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant.SchemeVectorConstant
attribute), 1357 lilypondfiletools.make_basic_lilypond_file()

lilypond_format (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff 1370
attribute), 1370 jad.tools.lilypondfiletools.make_basic_lilypond_file(),

lilypond_format (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff 1387
attribute), 1387 lilypondfiletools.make_floating_time_signature_lilypond_file()

lilypond_format (abjad.tools.scoretools.Score.Score.Score (in module ab-
attribute), 1395 jad.tools.lilypondfiletools.make_floating_time_signature_lilypond_file()),

lilypond_format (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup 1404
attribute), 1404 lilypondfiletools.make_time_signature_context_block()

lilypond_format (abjad.tools.skiptools.Skip.Skip.Skip at- (in module ab-
tribute), 1495 jad.tools.lilypondfiletools.make_time_signature_context_block()),

lilypond_format (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff 1655
attribute), 1655 lilypondfiletools.MIDIBlock (class in ab-

lilypond_format (abjad.tools.stafftools.Staff.Staff.Staff jad.tools.lilypondfiletools.MIDIBlock.MIDIBlock),
attribute), 1664 881

lilypond_format (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet, 1830
attribute), 1830 jad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock),

lilypond_format (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet 858
attribute), 1840 lilypondfiletools.PaperBlock (class in ab-

lilypond_format (abjad.tools.voicetools.Voice.Voice.Voice jad.tools.lilypondfiletools.PaperBlock.PaperBlock),
attribute), 1866 882

lilypondfiletools.AbjadRevisionToken (class in ab- lilypondfiletools.ScoreBlock (class in ab-
jad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken), 861 883
861 jad.tools.lilypondfiletools.ScoreBlock.ScoreBlock),

lilypondfiletools.AttributedBlock (class in ab- lilypondparsertools.GuileProxy (class in ab-

jad.tools.lilypondparsertools.GuileProxy.GuileProxy.local_maxima (abjad.tools.pitchtools.NamedChromaticPitchSegment.Name
 2034 attribute), 1215
 lilypondparsertools.lilypond_enharmonic_transpose() local_minima (abjad.tools.pitchtools.NamedChromaticPitchSegment.Name
 (in module ab- attribute), 1215
 jad.tools.lilypondparsertools.lilypond_enharmonic_transpose() abjad.tools.abctools.Parser.Parser.Parser at-
 2079 tribute), 1879
 lilypondparsertools.LilyPondDuration (class in ab- logger (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.Lily
 jad.tools.lilypondparsertools.LilyPondDuration.LilyPondDuration attribute), 2046
 2036 logger (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.
 lilypondparsertools.LilyPondEvent (class in ab- attribute), 2069
 jad.tools.lilypondparsertools.LilyPondEvent.LilyPondEvent abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.Scheme
 2038 attribute), 2074
 lilypondparsertools.LilyPondFraction (class in ab- logger (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.R
 jad.tools.lilypondparsertools.LilyPondFraction.LilyPondFraction attribute), 1344
 2039 logger_path (abjad.tools.abctools.Parser.Parser.Parser at-
 lilypondparsertools.LilyPondLexicalDefinition tribute), 1879
 (class in ab- logger_path (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.
 jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition),
 2040 logger_path (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyP
 lilypondparsertools.LilyPondParser (class in ab- attribute), 2069
 jad.tools.lilypondparsertools.LilyPondParser.LilyPondParser logger_path (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.S
 2044 attribute), 2074
 lilypondparsertools.LilyPondSyntacticalDefinition logger_path (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreePa
 (class in ab- attribute), 1344
 jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition abjad.tools.lilypondparsertools.ComplexBeamSpanner.ComplexBeamSpanner.
 2048 attribute), 232
 lilypondparsertools.parse_reduced_ly_syntax() lone (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComple
 (in module ab- attribute), 240
 jad.tools.lilypondparsertools.parse_reduced_ly_syntax() lone (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredCom
 2079 attribute), 249
 lilypondparsertools.ReducedLyParser (class in ab- long_description (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookS
 jad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser attribute), 1887
 2067 long_description (abjad.tools.developerscripttools.AbjDevScript.AbjDevScr
 lilypondparsertools.SchemeParser (class in ab- attribute), 1916
 jad.tools.lilypondparsertools.SchemeParser.SchemeParser long_description (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepS
 2074 attribute), 1918
 lilypondparsertools.SyntaxNode (class in ab- long_description (abjad.tools.developerscripttools.BuildApiScript.BuildApi
 jad.tools.lilypondparsertools.SyntaxNode.SyntaxNode), attribute), 1920
 2078 long_description (abjad.tools.developerscripttools.CleanScript.CleanScript.
 limit_denominator() (ab- attribute), 1923
 jad.tools.durationtools.Duration.Duration.Duration long_description (abjad.tools.developerscripttools.CountLinewidthsScript.C
 method), 446 attribute), 1925
 limit_denominator() (ab- long_description (abjad.tools.developerscripttools.CountToolsScript.CountT
 jad.tools.durationtools.Offset.Offset.Offset attribute), 1928
 method), 450 long_description (abjad.tools.developerscripttools.DeveloperScript.Develop
 limit_denominator() (ab- attribute), 1910
 jad.tools.mathtools.NonreducedFraction.NonreducedFraction long_description (abjad.tools.developerscripttools.DirectoryScript.Directory
 method), 949 attribute), 1912
 lines (abjad.tools.abjadbooktools.AbjadBookProcessor.AbjadBookProcessor long_description (abjad.tools.developerscripttools.MakeNewClassTemplate
 attribute), 1885 attribute), 1930
 lines (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock long_description (abjad.tools.developerscripttools.MakeNewFunctionTemp
 attribute), 1888 attribute), 1932
 lines (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner long_description (abjad.tools.developerscripttools.RenameModulesScript.R
 attribute), 1613 attribute), 1934

long_description (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript attribute), 1937

long_description (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript attribute), 1940

long_description (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript attribute), 1942

long_description (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript attribute), 1945

long_description (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript attribute), 1947

long_description (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript attribute), 1949

long_description (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript attribute), 1952

M

marktools.detach_lilypond_comments_attached_to_component() (in module abjad.tools.marktools.detach_lilypond_comments_attached_to_component), 916

make_performer_name_instrument_dictionary() (abjad.tools.scoretools.Performer.Performer method), 1381

makeClusters() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy method), 2035

mark (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner attribute), 1544

mark() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy method), 2035

marktools.Annotation (class in abjad.tools.marktools.Annotation), 890

marktools.Articulation (class in abjad.tools.marktools.Articulation), 893

marktools.attach_annotations_to_components_in_expr() (in module abjad.tools.marktools.attach_annotations_to_components_in_expr), 913

marktools.attach_articulations_to_notes_and_chords_in_expr() (in module abjad.tools.marktools.attach_articulations_to_notes_and_chords_in_expr), 914

marktools.attach_lilypond_command_marks_to_components_in_expr() (in module abjad.tools.marktools.attach_lilypond_command_marks_to_components_in_expr), 914

marktools.attach_lilypond_comments_to_components_in_expr() (in module abjad.tools.marktools.attach_lilypond_comments_to_components_in_expr), 914

marktools.attach_stem_tremolos_to_notes_and_chords_in_expr() (in module abjad.tools.marktools.attach_stem_tremolos_to_notes_and_chords_in_expr), 915

marktools.BarLine (class in abjad.tools.marktools.BarLine), 896

marktools.detach_marks_attached_to_component() (in module abjad.tools.marktools.detach_marks_attached_to_component), 918

marktools.detach_marks_attached_to_components_in_expr() (in module abjad.tools.marktools.detach_marks_attached_to_components_in_expr), 918

marktools.detach_noncontext_marks_attached_to_component() (in module abjad.tools.marktools.detach_noncontext_marks_attached_to_component), 919

marktools.detach_stem_tremolos_attached_to_component() (in module abjad.tools.marktools.detach_stem_tremolos_attached_to_component), 919

marktools.DirectedMark (class in abjad.tools.marktools.DirectedMark), 888

marktools.get_annotation_attached_to_component() (in module abjad.tools.marktools.get_annotation_attached_to_component), 920

marktools.get_annotations_attached_to_component() (in module abjad.tools.marktools.get_annotations_attached_to_component), 920

marktools.get_articulation_attached_to_component() (in module abjad.tools.marktools.get_articulation_attached_to_component), 921

marktools.get_articulations_attached_to_component() (in module abjad.tools.marktools.get_articulations_attached_to_component), 921

[illegible]

Index	2267
--------------	-------------

965
`mathtools.is_assignable_integer()` (in module `abjad.tools.mathtools.is_assignable_integer`), 965
`mathtools.is_dotted_integer()` (in module `abjad.tools.mathtools.is_dotted_integer`), 966
`mathtools.is_integer_equivalent_expr()` (in module `abjad.tools.mathtools.is_integer_equivalent_expr`), 967
`mathtools.is_integer_equivalent_number()` (in module `abjad.tools.mathtools.is_integer_equivalent_number`), 967
`mathtools.is_negative_integer()` (in module `abjad.tools.mathtools.is_negative_integer`), 967
`mathtools.is_nonnegative_integer()` (in module `abjad.tools.mathtools.is_nonnegative_integer`), 968
`mathtools.is_nonnegative_integer_equivalent_number()` (in module `abjad.tools.mathtools.is_nonnegative_integer_equivalent_number`), 968
`mathtools.is_nonnegative_integer_power_of_two()` (in module `abjad.tools.mathtools.is_nonnegative_integer_power_of_two`), 968
`mathtools.is_positive_integer()` (in module `abjad.tools.mathtools.is_positive_integer`), 969
`mathtools.is_positive_integer_equivalent_number()` (in module `abjad.tools.mathtools.is_positive_integer_equivalent_number`), 969
`mathtools.is_positive_integer_power_of_two()` (in module `abjad.tools.mathtools.is_positive_integer_power_of_two`), 969
`mathtools.least_common_multiple()` (in module `abjad.tools.mathtools.least_common_multiple`), 970
`mathtools.least_multiple_greater_equal()` (in module `abjad.tools.mathtools.least_multiple_greater_equal`), 970
`mathtools.least_power_of_two_greater_equal()` (in module `abjad.tools.mathtools.least_power_of_two_greater_equal`), 970
`mathtools.next_integer_partition()` (in module `abjad.tools.mathtools.next_integer_partition`), 971
`mathtools.NonreducedFraction` (class in `abjad.tools.mathtools.NonreducedFraction.NonreducedFraction`), 947
`mathtools.partition_integer_by_ratio()` (in module `abjad.tools.mathtools.partition_integer_by_ratio`), 972
`mathtools.partition_integer_into_canonic_parts()` (in module `abjad.tools.mathtools.partition_integer_into_canonic_parts`), 972
`mathtools.partition_integer_into_halves()` (in module `abjad.tools.mathtools.partition_integer_into_halves`), 973
`mathtools.partition_integer_into_units()` (in module `abjad.tools.mathtools.partition_integer_into_units`), 974
`mathtools.Ratio` (class in `abjad.tools.mathtools.Ratio.Ratio`), 953
`mathtools.remove_powers_of_two()` (in module `abjad.tools.mathtools.remove_powers_of_two`), 974
`mathtools.sign()` (in module `abjad.tools.mathtools.sign`), 975
`mathtools.weight()` (in module `abjad.tools.mathtools.weight`), 975
`mathtools.yield_all_compositions_of_integer()` (in module `abjad.tools.mathtools.yield_all_compositions_of_integer`), 975
`mathtools.yield_all_partitions_of_integer()` (in module `abjad.tools.mathtools.yield_all_partitions_of_integer`), 976
`measure_number` (`abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure` attribute), 980
`measure_number` (`abjad.tools.measuretools.DynamicMeasure.DynamicMeasure` attribute), 991
`measure_number` (`abjad.tools.measuretools.Measure.Measure.Measure` attribute), 1002
`measures` (`abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment` attribute), 1859
`measuretools.all_are_measures()` (in module `abjad.tools.measuretools.all_are_measures`), 1009
`measuretools.AnonymousMeasure` (class in `abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure`), 977
`measuretools.append_spacer_skip_to_underfull_measure()` (in module `abjad.tools.measuretools.append_spacer_skip_to_underfull_measure`), 1010
`measuretools.append_spacer_skips_to_underfull_measures_in_expr()` (in module `abjad.tools.measuretools.append_spacer_skips_to_underfull_measures_in_expr`), 1011
`measures.apply_full_measure_tuplets_to_contents_of_measures_in_expr()` (in module `abjad.tools.measuretools.apply_full_measure_tuplets_to_contents_of_measures_in_expr`), 1011

1011 (in module ab-
measuretools.comment_measures_in_container_with_measure_number(
(in module ab- 1020
jad.tools.measuretools.comment_measures_in_container_with_measure_number(
1012 (in module ab-
measuretools.DynamicMeasure (class in ab- jad.tools.measuretools.list_time_signatures_of_measures_in_expr(
jad.tools.measuretools.DynamicMeasure.DynamicMeasure) 1021
989 measuretools.make_measures_with_full_measure_spacer_skips(
measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets(
(in module ab- jad.tools.measuretools.make_measures_with_full_measure_spacer_skips(
jad.tools.measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets),
1013 measuretools.Measure (class in ab-
measuretools.fill_measures_in_expr_with_full_measure_spacer_skips(
(in module ab- 1000
jad.tools.measuretools.fill_measures_in_expr_with_full_measure_spacer_skips(
1014 (in module ab-
measuretools.fill_measures_in_expr_with_minimal_number_of_notes(
(in module ab- 1022
jad.tools.measuretools.fill_measures_in_expr_with_minimal_number_of_notes(
1014 (in module ab-
measuretools.fill_measures_in_expr_with_repeated_notes(
(in module ab- 1023
jad.tools.measuretools.fill_measures_in_expr_with_repeated_notes(
1015 (in module ab-
measuretools.fill_measures_in_expr_with_time_signature_denominator(
(in module ab- 1023
jad.tools.measuretools.fill_measures_in_expr_with_time_signature_denominator(
1015 (in module ab-
measuretools.fuse_contiguous_measures_in_container_cyclically_by_jadtools(
(in module ab- 1024
jad.tools.measuretools.fuse_contiguous_measures_in_container_cyclically_by_jadtools(
1015 (in module ab-
measuretools.fuse_measures() (in module ab- jad.tools.measuretools.multiply_contents_of_measures_in_expr(
jad.tools.measuretools.fuse_measures), 1017 1024
measuretools.get_first_measure_in_improper_parentage_of_measures_in_expr(
(in module ab- (in module ab-
jad.tools.measuretools.get_first_measure_in_improper_parentage_of_measures_in_expr(
1018 1025
measuretools.get_first_measure_in_proper_parentage_of_measures_in_expr(
(in module ab- (in module ab-
jad.tools.measuretools.get_first_measure_in_proper_parentage_of_measures_in_expr(
1018 1027
measuretools.get_next_measure_from_component() measuretools.replace_contents_of_measures_in_expr(
(in module ab- (in module ab-
jad.tools.measuretools.get_next_measure_from_component(
1018 1028
measuretools.get_nth_measure_in_expr() (in module ab- measuretools.report_time_signature_distribution(
jad.tools.measuretools.get_nth_measure_in_expr), (in module ab-
1019 jad.tools.measuretools.report_time_signature_distribution),
measuretools.get_one_indexed_measure_number_in_expr() 1029
(in module ab- measuretools.scale_contents_of_measures_in_expr(
jad.tools.measuretools.get_one_indexed_measure_number_in_expr(
1020 (in module ab-
measuretools.get_previous_measure_from_component() 1030

measuretools.scale_measure_and_adjust_time_signature() melodic_chromatic_intervals (ab-
(in module ab- jad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChrom
jad.tools.measuretools.scale_measure_and_adjust_time_signature()), 1186
1030 melodic_counterpoint_interval (ab-
measuretools.scale_measure_denominator_and_adjust_measure_content() jad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInte
(in module ab- attribute), 1193
jad.tools.measuretools.scale_measure_denominator_and_adjust_measure_content() melodic_counterpoint_interval_class (ab-
1030 jad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounter
measuretools.set_always_format_time_signature_of_measures_in_explicit() (attribute), 1189
(in module ab- melodic_diatonic_interval_ascending (ab-
jad.tools.measuretools.set_always_format_time_signature_of_measures_in_explicit() jad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicI
1031 attribute), 1146
measuretools.set_measure_denominator_and_adjust_numerators() melodic_diatonic_interval_class (ab-
(in module ab- jad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInte
jad.tools.measuretools.set_measure_denominator_and_adjust_numerators() attribute), 1193
1031 melodic_diatonic_interval_class_segment (ab-
mediant (abjad.tools.tonalitytools.Scale.Scale.Scale at- jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChrom
tribute), 1816 attribute), 1216
melodic_chromatic_interval (ab- melodic_diatonic_interval_descending (ab-
jad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval.HarmonicDiatonicI
attribute), 1193 attribute), 1146
melodic_chromatic_interval_class (ab- melodic_diatonic_interval_numbers (ab-
jad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticIntervalSet.MelodicDiatonicI
attribute), 1175 attribute), 1200
melodic_chromatic_interval_class_segment (ab- melodic_diatonic_interval_segment (ab-
jad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment
attribute), 1183 attribute), 1152
melodic_chromatic_interval_class_segment (ab- melodic_diatonic_interval_segment (ab-
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment
attribute), 1216 attribute), 1216
melodic_chromatic_interval_class_vector (ab- melodic_diatonic_interval_segment (ab-
jad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment
attribute), 1183 attribute), 1808
melodic_chromatic_interval_numbers (ab- melodic_diatonic_interval_segment (ab-
jad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment
attribute), 1183 attribute), 1812
melodic_chromatic_interval_numbers (ab- melodic_diatonic_intervals (ab-
jad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicDiatonicI
attribute), 1186 attribute), 1200
melodic_chromatic_interval_segment (ab- merge() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchA
jad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment
attribute), 1152 meters (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.Me
melodic_chromatic_interval_segment (ab- attribute), 1579
jad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment
attribute), 1197 attribute), 1959
melodic_chromatic_interval_segment (ab- middle_c_position (ab-
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment
attribute), 1216 attribute), 378
melodic_chromatic_interval_segment (ab- minimal_page_breaking (ab-
jad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator
attribute), 1807 attribute), 883
melodic_chromatic_interval_set (ab- mode (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeyS
jad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet
attribute), 1200 mode_name (abjad.tools.tonalitytools.Mode.Mode.Mode

attribute), 1812
 module_crawler (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler
 attribute), 1954
 module_crawler (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler
 attribute), 1957
 module_crawler (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler
 attribute), 1962
 module_name (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter
 attribute), 1959
 module_name (abjad.tools.documentationtools.Documenter.Documenter.Documenter
 attribute), 1960
 module_name (abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter
 attribute), 1964
 modulo (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass
 attribute), 1490
 most_likely_instrument_based_on_performer_name (ab-
 jad.tools.scoretools.Performer.Performer.Performer
 attribute), 1380
 multiplied_duration (ab-
 jad.tools.chordtools.Chord.Chord.Chord
 attribute), 267
 multiplied_duration (abjad.tools.leaftools.Leaf.Leaf.Leaf
 attribute), 828
 multiplied_duration (ab-
 jad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic
 attribute), 1033
 multiplied_duration (ab-
 jad.tools.notetools.Note.Note.Note
 attribute), 1038
 multiplied_duration (ab-
 jad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest
 attribute), 1308
 multiplied_duration (abjad.tools.resttools.Rest.Rest.Rest
 attribute), 1311
 multiplied_duration (abjad.tools.skiptools.Skip.Skip.Skip
 attribute), 1495
 multiplied_duration (ab-
 jad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet
 attribute), 1831
 multiplied_duration (ab-
 jad.tools.tuplettools.Tuplet.Tuplet.Tuplet
 attribute), 1841
 multiplier (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark
 attribute), 419
 multiplier (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure
 attribute), 980
 multiplier (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure
 attribute), 992
 multiplier (abjad.tools.measuretools.Measure.Measure.Measure
 attribute), 1003
 multiplier (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet
 attribute), 1834
 multiplier (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet
 attribute), 1843
 multiply() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass
 attribute), 1331
 multiply() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment
 attribute), 1337
 multiply() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet
 attribute), 1367
 multiply() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow
 attribute), 1367
 music (abjad.tools.abctools.ScoreSelection.ScoreSelection.ScoreSelection
 attribute), 1881
 music (abjad.tools.containertools.Cluster.Cluster.Cluster
 attribute), 336
 music (abjad.tools.containertools.Container.Container.Container
 attribute), 343
 music (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer
 attribute), 350
 music (abjad.tools.contexttools.Context.Context.Context
 attribute), 385
 music (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer
 attribute), 475
 music (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure
 attribute), 981
 music (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure
 attribute), 992
 music (abjad.tools.measuretools.Measure.Measure.Measure
 attribute), 1003
 music (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff
 attribute), 1370
 music (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff
 attribute), 1387
 music (abjad.tools.scoretools.Score.Score.Score
 attribute), 1395
 music (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup
 attribute), 1404
 music (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff
 attribute), 1655
 music (abjad.tools.stafftools.Staff.Staff.Staff
 attribute), 1661
 music (abjad.tools.tietools.TieChain.TieChain.TieChain
 attribute), 1683
 music (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet
 attribute), 1831
 music (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet
 attribute), 1841
 music (abjad.tools.verticaltools.VerticalMoment.VerticalMoment.VerticalMoment
 attribute), 1859
 music (abjad.tools.voicetools.Voice.Voice.Voice
 attribute), 1866
 N
 name (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory
 attribute), 381
 name (abjad.tools.contexttools.Context.Context.Context
 attribute), 387

name (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark (ab-
attribute), 403
name (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory
attribute), 415
name (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory (ab-
attribute), 1905
name (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory (ab-
attribute), 659
name (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock), 1209
attribute), 870
name (abjad.tools.marktools.Annotation.Annotation.Annotation jad.tools.tonalitytools.Scale.Scale.Scale at-
attribute), 891
name (abjad.tools.marktools.Articulation.Articulation.Articulation named_chromatic_pitch_class_to_scale_degree()
attribute), 894
name (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory (ab-
attribute), 937
name (abjad.tools.pitchtools.Accidental.Accidental.Accidental jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChrom
attribute), 1130
name (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping
attribute), 1251
name (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMapping
attribute), 1257
name (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory NamedChromaticPitchClassSet.NamedChrom
attribute), 1262
name (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff named_chromatic_pitch_classes (ab-
attribute), 1372
name (abjad.tools.scoretools.Performer.Performer.Performer jad.tools.tonalitytools.ChordClass.ChordClass.ChordClass
attribute), 1381
name (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory named_chromatic_pitch_classes (ab-
attribute), 1382
name (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff named_chromatic_pitch_set (ab-
attribute), 1389
name (abjad.tools.scoretools.Score.Score.Score at- attribute), 1216
attribute), 1398
name (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChrom
attribute), 1407
name (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff named_chromatic_pitches (ab-
attribute), 1658
name (abjad.tools.stafftools.Staff.Staff.Staff attribute), attribute), 1216
1667
name (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator jad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticP
attribute), 1809
name (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator named_chromatic_pitches (ab-
attribute), 1811
name (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree attribute), 1221
attribute), 1819
name (abjad.tools.voicetools.Voice.Voice.Voice at- jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch
attribute), 1869
named_chromatic_pitch (ab- named_diatonic_pitch (ab-
jad.tools.notetools.NoteHead.NoteHead.NoteHead jad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPi
attribute), 1041
named_chromatic_pitch (ab- named_diatonic_pitch_class (ab-
jad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch NamedChromaticPitch.NamedChromaticPitch
attribute), 1225

named_diatonic_pitch_class (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch attribute), 1226

named_diatonic_pitch_class (abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch attribute), 1247

named_diatonic_pitch_class (abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass attribute), 1249

negative_level (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1425

negative_level (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1438

next (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell attribute), 1053

next_vertical_moment (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment attribute), 1859

nodes (abjad.tools.datastructuretools.Digraph.Digraph.Digraph attribute), 1900

normalized_spacing_duration (abjad.tools.layouttools.SpacingIndication.SpacingIndication attribute), 824

note_head (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic attribute), 1034

note_head (abjad.tools.notetools.Note.Note.Note attribute), 1039

note_heads (abjad.tools.chordtools.Chord.Chord.Chord attribute), 268

notes (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment attribute), 1859

notetools.add_artificial_harmonic_to_note() (in module abjad.tools.notetools.add_artificial_harmonic_to_note), 1043

notetools.all_are_notes() (in module abjad.tools.notetools.all_are_notes), 1044

notetools.make_accelerating_notes_with_lilypond_multipliers() (in module abjad.tools.notetools.make_accelerating_notes_with_lilypond_multipliers), 1044

notetools.make_notes() (in module abjad.tools.notetools.make_notes), 1045

notetools.make_notes_with_multiplied_durations() (in module abjad.tools.notetools.make_notes_with_multiplied_durations), 1046

notetools.make_percussion_note() (in module abjad.tools.notetools.make_percussion_note), 1046

notetools.make_quarter_notes_with_lilypond_multipliers() (in module abjad.tools.notetools.make_quarter_notes_with_lilypond_multipliers), 1046

notetools.make_repeated_notes() (in module abjad.tools.notetools.make_repeated_notes), 1047

notetools.make_repeated_notes_from_time_signature() (in module abjad.tools.notetools.make_repeated_notes_from_time_signature), 1047

notetools.make_repeated_notes_with_shorter_notes_at_end() (in module abjad.tools.notetools.make_repeated_notes_with_shorter_notes_at_end), 1048

notetools.make_repeated_notes_with_shorter_notes_at_end() (in module abjad.tools.notetools.make_repeated_notes_with_shorter_notes_at_end), 1048

notetools.make_tied_note() (in module abjad.tools.notetools.make_tied_note), 1049

notetools.NaturalHarmonic (class in abjad.tools.notetools.NaturalHarmonic), 1032

notetools.Note (class in abjad.tools.notetools.Note), 1037

notetools.NoteHead (class in abjad.tools.notetools.NoteHead), 1041

notetools.yield_groups_of_notes_in_sequence() (in module abjad.tools.notetools.yield_groups_of_notes_in_sequence), 1049

number (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject attribute), 1064

number (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject attribute), 1070

number (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject attribute), 1072

number (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject attribute), 1075

number (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject attribute), 1077

number (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval attribute), 1131

number (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass attribute), 1133

number (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval attribute), 1143

number (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass attribute), 1144

number (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval attribute), 1146

number (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass attribute), 1147

number (abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject attribute), 1148

attribute), 1082	numbered_chromatic_pitch_class_segment	(ab-
number (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.HarmonicIntervalObject), 1084	jad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject.HarmonicIntervalObject	attribute), 1817
number (abjad.tools.pitchtools.IntervalObject.IntervalObject.IntervalObject), 1092	numbered_chromatic_pitch_class_set	(ab-
number (abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass.IntervalObjectClass), 1093	jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment	attribute), 1817
number (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass), 1158	numbered_chromatic_pitch_class_set	(ab-
number (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass), 1167	jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment	attribute), 1212
number (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval), 1175	numbered_chromatic_pitch_class_set	(ab-
number (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass), 1176	jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment	attribute), 1212
number (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval), 1189	numbered_chromatic_pitch_class_set	(ab-
number (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass), 1191	jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment	attribute), 1212
number (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval), 1193	numbered_chromatic_pitch_class_set	(ab-
number (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass), 1195	jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment	attribute), 1212
number (abjad.tools.pitchtools.MelodicIntervalClassObject.MelodicIntervalClassObject.MelodicIntervalClassObject), 1100	numbered_chromatic_pitch_class_set	(ab-
number (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject.MelodicIntervalObject), 1102	jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment	attribute), 1212
number (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator), 1809	numbered_chromatic_pitch_class_set	(ab-
number (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator), 1811	jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment	attribute), 1212
number (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree), 1819	numbered_chromatic_pitch_class_set	(ab-
numbered_chromatic_pitch	numbered_chromatic_pitch_classes	(ab-
jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch), 1205	jad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet	attribute), 1240
numbered_chromatic_pitch	numbered_chromatic_pitch_classes	(ab-
jad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch), 1226	jad.tools.pitchtools.NamedChromaticPitchClassVector.NamedChromaticPitchClassVector	attribute), 1244
numbered_chromatic_pitch_class	numbered_chromatic_pitch_classes	(ab-
jad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.NamedChromaticPitchClass), 1207	jad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet	attribute), 1240
numbered_chromatic_pitch_class	numbered_chromatic_pitch_classes	(ab-
jad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.NamedChromaticPitchClass), 1207	jad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet	attribute), 1240
numbered_chromatic_pitch_class	numbered_chromatic_pitch_classes	(ab-
jad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch), 1226	jad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet	attribute), 1240
numbered_chromatic_pitch_class_segment	numbered_chromatic_pitch_classes	(ab-
jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment), 1209	jad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet	attribute), 1240
numbered_chromatic_pitch_class_segment	numbered_chromatic_pitch_classes	(ab-
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment), 1216	jad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet	attribute), 1240

[numbered_diatonic_pitch_class](#) (abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass.NamedDiatonicPitchClass attribute), [1228](#)
[numbered_diatonic_pitch_class](#) (abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch.NumberedDiatonicPitch attribute), [1247](#)
[numerator](#) (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark attribute), [420](#)
[numerator](#) (abjad.tools.durationtools.Duration.Duration.Duration attribute), [446](#)
[numerator](#) (abjad.tools.durationtools.Offset.Offset.Offset attribute), [450](#)
[numerator](#) (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction.NonreducedFraction attribute), [948](#)
O
[object](#) (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter attribute), [1959](#)
[object](#) (abjad.tools.documentationtools.Documenter.Documenter.Documenter attribute), [1960](#)
[object](#) (abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter attribute), [1964](#)
[octave_number](#) (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch attribute), [1205](#)
[offset](#) (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer attribute), [1325](#)
[offset](#) (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf attribute), [1339](#)
[offset](#) (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode attribute), [1318](#)
[offset_counts](#) (abjad.tools.timeintervaltools.TimeIntervalAggregateMaxim.TimeIntervalAggregateMaxim.TimeIntervalAggregateMaxim attribute), [1701](#)
[offset_counts](#) (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree attribute), [1712](#)
[offset_counts](#) (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), [1725](#)
[offsets](#) (abjad.tools.timeintervaltools.TimeIntervalAggregateMaxim.TimeIntervalAggregateMaxim.TimeIntervalAggregateMaxim attribute), [1701](#)
[offsets](#) (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree attribute), [1712](#)
[offsets](#) (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), [1725](#)
[offsettools.update_offset_values_of_component\(\)](#) (in module abjad.tools.offsettools.update_offset_values_of_component), [2080](#)
[offsettools.update_offset_values_of_component_in_seconds\(\)](#) (in module abjad.tools.offsettools.update_offset_values_of_component_in_seconds), [2080](#)
[one_line_named_chromatic_pitch_repr](#) (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), [1260](#)
[one_line_numbered_chromatic_pitch_repr](#) (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), [1260](#)
[one_voice\(\)](#) (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy attribute), [2035](#)
[operator](#) (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression attribute), [1459](#)
[order_by\(\)](#) (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet attribute), [1213](#)
[order_by\(\)](#) (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass attribute), [1805](#)
[output_format](#) (abjad.tools.abjadbooktools.AbjadBookProcessor.AbjadBookProcessor.AbjadBookProcessor attribute), [1885](#)
[output_formats](#) (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.AbjadBookScript attribute), [1887](#)
[output_path](#) (abjad.tools.abctools.Parser.Parser.Parser attribute), [1879](#)
[output_path](#) (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser attribute), [2046](#)
[output_path](#) (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser attribute), [2070](#)
[output_path](#) (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser attribute), [2074](#)
[output_path](#) (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser attribute), [1344](#)
[overlap_components](#) (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer attribute), [1325](#)
[overlap_leaves](#) (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf attribute), [1339](#)
[overlap_leaves](#) (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode attribute), [1318](#)
[overlap_measures](#) (abjad.tools.timeintervaltools.TimeIntervalAggregateMaxim.TimeIntervalAggregateMaxim.TimeIntervalAggregateMaxim attribute), [1860](#)
[overlap_notes](#) (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree attribute), [1860](#)
[override](#) (abjad.tools.beamttools.BeamSpanner.BeamSpanner.BeamSpanner attribute), [224](#)
[override](#) (abjad.tools.beamttools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner attribute), [231](#)
[override](#) (abjad.tools.beamttools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), [239](#)
[override](#) (abjad.tools.beamttools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), [248](#)
[override](#) (abjad.tools.beamttools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner attribute), [256](#)
[override](#) (abjad.tools.chordtools.Chord.Chord.Chord attribute), [267](#)
[override](#) (abjad.tools.componenttools.Component.Component.Component attribute), [276](#)
[override](#) (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), [336](#)
[override](#) (abjad.tools.containertools.Container.Container.Container attribute), [343](#)
[override](#) (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer attribute), [350](#)

override (abjad.tools.contexttools.Context.Context.Context attribute), 385
 override (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer attribute), 476
 override (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 828
 override (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock attribute), 869
 override (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 981
 override (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 992
 override (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1003
 override (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic attribute), 1033
 override (abjad.tools.notetools.Note.Note.Note attribute), 1038
 override (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest attribute), 1308
 override (abjad.tools.resttools.Rest.Rest.Rest attribute), 1311
 override (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1370
 override (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1387
 override (abjad.tools.scoretools.Score.Score.Score attribute), 1396
 override (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1405
 override (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1496
 override (abjad.tools.spannertools.BracketSpanner.BracketSpanner attribute), 1513
 override (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner attribute), 1520
 override (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner attribute), 1528
 override (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner attribute), 1536
 override (abjad.tools.spannertoolsDirectedSpannerDirectedSpanner attribute), 1501
 override (abjad.tools.spannertoolsDynamicTextSpannerDynamicTextSpanner attribute), 1544
 override (abjad.tools.spannertoolsGlissandoSpannerGlissandoSpanner attribute), 1550
 override (abjad.tools.spannertoolsHairpinSpannerHairpinSpanner attribute), 1557
 override (abjad.tools.spannertoolsHiddenStaffSpannerHiddenStaffSpanner attribute), 1565
 override (abjad.tools.spannertoolsHorizontalBracketSpannerHorizontalBracketSpanner attribute), 1571
 override (abjad.tools.spannertoolsMetricGridSpannerMetricGridSpanner attribute), 1578
 override (abjad.tools.spannertoolsOctavationSpannerOctavationSpanner attribute), 1585
 override (abjad.tools.spannertoolsPhrasingSlurSpannerPhrasingSlurSpanner attribute), 1592
 override (abjad.tools.spannertoolsPianoPedalSpannerPianoPedalSpanner attribute), 1598
 override (abjad.tools.spannertoolsSlurSpannerSlurSpanner attribute), 1605
 override (abjad.tools.spannertoolsSpannerSpanner attribute), 1507
 override (abjad.tools.stafftools.StaffLinesSpanner.StaffLinesSpanner attribute), 1612
 override (abjad.tools.spannertoolsTextScriptSpannerTextScriptSpanner attribute), 1619
 override (abjad.tools.spannertoolsTextSpannerTextSpanner attribute), 1625
 override (abjad.tools.spannertoolsTrillSpannerTrillSpanner attribute), 1632
 override (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1655
 override (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1664
 override (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute), 1685
 override (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1831
 override (abjad.tools.tuplettoolsTupletTupletTuplet attribute), 1841
 override (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1866

P

p_apostrophes__apostrophes__APOSTROPHE() (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser method), 2070
 p_assignment__assignment__P_ASSIGNMENT() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2049
 p_assignment__embedded_scm() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2049
 p_assignment__id__P_ASSIGNMENT() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2049
 p_bare_number__bare_number_closed() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2049
 p_bare_number__REAL__NUMBER_IDENTIFIER() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2049
 p_bare_number__UNSIGNED__NUMBER_IDENTIFIER() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2049

p_bare_number_closed__NUMBER_IDENTIFIER()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_bare_number_closed__REAL()	(ab- jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_bare_number_closed__UNSIGNED()	(ab- jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_beam__BRACKET_L()	(ab- jad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser. method), 2070
p_beam__BRACKET_R()	(ab- jad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser. method), 2070
p_boolean__BOOLEAN()	(ab- jad.tools.lilypondparsertools.SchemeParser.SchemeParser. method), 2075
p_braced_music_list__Chr123__music_list__Chr125()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_chord_body__ANGLE_OPEN__chord_body_elements__ANGLE_CLOSE__prefix__CONTEXT__simple_string_optional_id_opt	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_chord_body__chord_pitches()	(ab- jad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser. method), 2070
p_chord_body__chord_pitches__positive_leaf_duration()	(abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser. method), 2070
p_chord_body_element__music_function_chord_body()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_chord_body_element__pitch_exclamations_questions_punctuation_leads_events()	(ab- jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_chord_body_elements__chord_body_elements__chord_body_element__tuple()	(ab- jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_chord_body_elements_Empty()	(ab- jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_chord_pitches__CARAT_L__pitches__CARAT_R()	(ab- jad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser. method), 2070
p_closed_music__complex_music_prefix__closed_music()	(ab- jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049
p_closed_music__music_bare()	(ab- jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition. method), 2049

method), 2051

p_direction_less_char__E_CLOSE() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_direction_less_char__E_EXCLAMATION() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_direction_less_char__E_OPEN() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_direction_less_event__direction_less_char() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_direction_less_event__event_function_event() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_direction_less_event__EVENT_IDENTIFIER() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_direction_less_event__tremolo_type() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_direction_reqd_event__gen_text_def() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_direction_reqd_event__script_abbreviation() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_dots__dots__Chr46() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_dots__dots__DOT() (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser method), 2071

p_dots__Empty() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2051

p_dots__EMPTY() (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser method), 2071

p_duration_length__multiplied_duration() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm__embedded_scm_bare() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm__scm_function_call() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_arg__embedded_scm_bare_arg() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_arg__music_arg() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_arg__scm_function_call() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_arg__closed__closed_music() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_arg__closed__embedded_scm_bare_arg() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_arg__closed__scm_function_call_closed() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare__SCM_IDENTIFIER() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare__SCM_TOKEN() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare_arg__context_def_spec_block() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare_arg__context_modification() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare_arg__embedded_scm_bare() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare_arg__full_markup() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare_arg__full_markup_list() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare_arg__output_def() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare_arg__score_block() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare_arg__STRING() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_bare_arg__STRING_IDENTIFIER() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2052

p_embedded_scm_chord_body__bare_number() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2053

p_embedded_scm_chord_body__chord_body_element() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2053

p_embedded_scm_chord_body__embedded_scm_bare_arg() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition method), 2053

[illegible]

(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_common__function_arglist_common_minus__function_arglist_optional__function_arglist_backup__BACKUP() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_common_minus__EXPECT_SCM__function_arglist_optional__Chr45__NUMBER__IDENTIFIER() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_common_minus__EXPECT_SCM__function_arglist_optional__EXPECT_OPTIONAL__EXPECT_DURATION__ (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_common_minus__EXPECT_SCM__function_arglist_optional__EXPECT_OPTIONAL__EXPECT_DURATION__EXPECT_PITCH__function_arglist_optional__EXPECT_DURATION__EXPECT_PITCH__ (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_common_minus__function_arglist_common__function_arglist_optional__EXPECT_SCM__function_arglist_optional__EXPECT_DURATION__EXPECT_PITCH__ (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_keep__function_arglist_backup() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_keep__function_arglist_common() p_gen_text_def__full_markup() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_closed__duration_length() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_closed__duration_length()also_in_chords() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_closed__duration_length()scm_argclosed() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_closed__duration_length()bare_numberclosed() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_closed__duration_length()Chr45__NUMBER__IDENTIFIER() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_closed__duration_length()Chr45__REAL() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_closed__duration_length()UNSIGNED() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_closed__duration_length()FRACTION() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2056

p_function_arglist_nonbackup__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_closed__duration_length()arglist_closed__post_event__abfinger() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2057

p_function_arglist_optional__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_optional() (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition method), 2057

p_function_arglist_optional__EXPECT_OPTIONAL__EXPECT_DURATION__function_arglist_optional() (ab-

jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition	(method), 2058
p_identifier_init_post_event_nofinger()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2058
p_markup_simple_markup()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2058
p_identifier_init_score_block()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2058
p_markup_braced_list_Chrl23_markup_braced_list_body_Chrl25()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2058
p_identifier_init_string()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2058
p_markup_braced_list_body_Empty()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2058
p_leaf_INTEGER()	(abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser) (method), 1345
p_markup_braced_list_body_markup_braced_list_body_markup()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2058
p_markup_braced_list_body_markup_list()	(abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser) (method), 2071
p_markup_command_basic_arguments_EXPECT_MARKUP_LIST_markup()	(abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser) (method), 2058
p_markup_command_basic_arguments_EXPECT_NO_MORE_ARGS()	(abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser) (method), 2058
p_markup_command_basic_arguments_EXPECT_SCM_markup_command()	(abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser) (method), 2058
p_markup_command_list_MARKUP_LIST_FUNCTION_markup_command()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2059
p_markup_command_list_arguments_EXPECT_MARKUP_markup_command()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2059
p_markup_command_list_arguments_markup_command_basic_argument()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2059
p_markup_composed_list_markup_head_1_list_markup_braced_list()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2059
p_markup_header_HEADER_Chrl23_lilypond_header_body_markup_Chrl25()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2058
p_markup_head_1_list_markup_head_1_item()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2059
p_markup_head_1_list_markup_head_1_list_markup_head_1_item()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2059
p_markup_list_markup_braced_list()	(abjad.tools.lilypondparsertools.SchemeParser.SchemeParser) (method), 2076
p_markup_list_markup_command_list()	(abjad.tools.lilypondparsertools.SchemeParser.SchemeParser) (method), 2059
p_markup_list_markup_composed_list()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition) (method), 2059

p_method(), 2072			(method), 2064
p_scalar__bare_number()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_script_dir_Chr45()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_scalar__embedded_scm_arg()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_script_dir_Chr94()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_scalar_closed__bare_number()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_script_dir_Chr95()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_scalar_closed__embedded_scm_arg_closed()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_sequential_music__braced_music_list()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_scm_function_call__SCM_FUNCTION__function_arglist()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_sequential_music__SEQUENTIAL__braced_music_list()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_scm_function_call_closed__SCM_FUNCTION__function_arglist_closed()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_element__elements__simple_element()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_score_block__SCORE__Chr123__score_body__Chr125()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_element_pitch_exclamations_questions_octave_check_optimize()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_score_body__music()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_element_RESTNAME_optional_notemode_duration()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_score_body__score_body__lilypond_header()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_markup_MARKUP_FUNCTION_markup_command_basic_arguments()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_score_body__score_body__output_def()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_markup_MARKUP_IDENTIFIER()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_score_body__SCORE_IDENTIFIER()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_markup_markup_scm_MARKUP_IDENTIFIER()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_script_abbreviation__ANGLE_CLOSE()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_markup_SCORE__Chr123__score_body__Chr125()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_script_abbreviation__Chr124()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_markup_STRING()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_script_abbreviation__Chr43()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_markup_STRING_IDENTIFIER()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_script_abbreviation__Chr45()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_music_context_change()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2063			(method), 2064
p_script_abbreviation__Chr46()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_music_event_chord()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2064			(method), 2064
p_script_abbreviation__Chr94()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_music_music_property_def()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)
p_method(), 2064			(method), 2064
p_script_abbreviation__Chr95()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)	p_simple_music_property_def_OVERRIDE_context_prop_spec_properties()	(abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition)

method), 2066

p_tie__TILDE() (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser method), 2072

p_toplevel__EMPTY() (ab- attribute), 948

jad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser (ab- attribute), 1234

p_toplevel__toplevel__container() (ab- parent (abjad.tools.chordtools.Chord.Chord.Chord attribute), 267

jad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser (ab- parent (abjad.tools.componenttools.Component.Component.Component attribute), 276

p_toplevel_expression__composite_music() (ab- attribute), 276

jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition attribute), 336

p_toplevel_expression__full_markup() (ab- parent (abjad.tools.containertools.Container.Container.Container attribute), 342

jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition attribute), 342

p_toplevel_expression__full_markup_list() (ab- attribute), 350

jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition attribute), 385

p_toplevel_expression__lilypond_header() (ab- parent (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer attribute), 385

jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition attribute), 385

p_toplevel_expression__output_def() (ab- 829

jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition attribute), 981

p_toplevel_expression__score_block() (ab- parent (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 992

jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition attribute), 992

p_tremolo_type__Chr58() (ab- attribute), 1003

jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition attribute), 1033

p_tremolo_type__Chr58__bare_unsigned() (ab- parent (abjad.tools.notetools.Note.Note.Note attribute), 1033

jad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition attribute), 1033

p_tuplet__FRACTION__container() (ab- attribute), 1309

jad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser attribute), 1311

p_variable__IDENTIFIER() (ab- parent (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer attribute), 1326

jad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser attribute), 1326

p_vector__HASH__L_PAREN__data__R_PAREN() (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser.SchemeParser attribute), 1319

package_prefix (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator attribute), 1370

pad_to_depth() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1387

pad_to_width() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1396

pad_to_width() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 1405

pair (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark attribute), 1496

pair (abjad.tools.durationtools.Duration.Duration.Duration (abjad.tools.durationtools.Duration.Duration.Duration attribute), 1655

parent (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1664

parent (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1831

parent (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1841

parent (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1867

parent_array (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell attribute), 1053

parent_array (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn attribute), 1055

parent_array (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow attribute), 1057

parent_column (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell attribute), 1053

parent_graph (abjad.tools.datastructuretools.Digraph.Digraph.Digraph attribute), 1901

parent_row (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell attribute), 1053

parentage_ratios (abjad.tools.rhythmtreetools.RhythmTreeCpitch.RhythmTreeCpitch attribute), 1326

parentage_ratios (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1339

parentage_ratios (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode attribute), 1319

parser (abjad.tools.abctools.Parser.Parser.Parser attribute), 1879

parser (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser attribute), 2046

parser (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser attribute), 2070

parser (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser attribute), 2074

parser (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser attribute), 1344

parser_rules_object (abjad.tools.abctools.Parser.Parser.Parser attribute), 1879

parser_rules_object (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser attribute), 2046

parser_rules_object (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser attribute), 2070

parser_rules_object (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser attribute), 2074

parser_rules_object (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser attribute), 1344

partial (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark attribute), 420

partition() (abjad.tools.datastructuretools.Digraph.Digraph.Digraph attribute), 1901

performer_count (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier attribute), 1378

performer_name_string (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier attribute), 1378

performers (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier attribute), 1378

pitch (abjad.tools.pitchtools.Pitch.Pitch attribute), 1234

pitch_array (abjad.tools.pitcharraytools.PitchArray.PitchArray attribute), 1053

pitch_array (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn attribute), 1055

pitch_array (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow attribute), 1057

pitch_array_cell (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell attribute), 1053

pitch_array_column (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn attribute), 1055

pitch_array_row (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow attribute), 1057

pitch_class (abjad.tools.pitchtools.PitchClass.PitchClass attribute), 1234

pitch_class_color (abjad.tools.pitchtools.NumberedChromaticPitchClassColor.NumberedChromaticPitchClassColor attribute), 1234

pitch_iterables (abjad.tools.pitchtools.NumberedChromaticPitchClassColor.NumberedChromaticPitchClassColor attribute), 1234

pitch_range (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 519

pitch_range (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 525

pitch_range (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 537

pitch_range (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 543

pitch_range (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone attribute), 549

pitch_range (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice attribute), 555

pitch_range (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 561

pitch_range (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 561

pitch_range (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 561

pitch_range (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 561

pitch_range (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 561

pitch_range (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 561

pitch_range (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 561

pitch_range (abjad.tools.instrumenttools.Cello.Cello.Cello attribute), 561

pitch_range (abjad.tools.instrumenttools.ClarineteInA.ClarinetInA.ClarinetInA), 567	pitch_range (abjad.tools.instrumenttools.Tuba.Tuba.Tuba), 731
pitch_range (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass), 579	pitch_range (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion), 743
pitch_range (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet), 585	pitch_range (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone), 749
pitch_range (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute), 591	pitch_range (abjad.tools.instrumenttools.Viola.Viola.Viola), 755
pitch_range (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon), 603	pitch_range (abjad.tools.instrumenttools.Violin.Violin.Violin), 761
pitch_range (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone), 597	pitch_range (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone), 767
pitch_range (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice), 609	pitch_range (abjad.tools.pitchtools.PitchArrayRow.PitchArrayRow.PitchArrayRow), 1057
pitch_range (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet), 615	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn), 621	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Flute.Flute.Flute), 627	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn), 632	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel), 638	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Guitar.Guitar.Guitar), 644	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Harp.Harp.Harp), 650	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord), 656	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Marimba.Marimba.Marimba), 665	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice), 671	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Oboe.Oboe.Oboe), 677	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Piano.Piano.Piano), 683	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo), 689	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone), 695	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone), 701	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice), 707	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone), 713	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone), 719	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice), 725	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260
pitch_range (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet), 731	pitch_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange), 1260

jad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell), (in module ab-
 1052 jad.tools.pitchtools.calculate_harmonic_diatonic_interval),
 pitcharraytools.PitchArrayColumn (class in ab- 1270
 jad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn),
 1054 jad.tools.pitchtools.calculate_harmonic_diatonic_interval_class()
 pitcharraytools.PitchArrayRow (class in ab- (in module ab-
 jad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow), 1270
 1056 pitchtools.calculate_melodic_chromatic_interval()
 pitches (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray (in module ab-
 attribute), 1050 jad.tools.pitchtools.calculate_melodic_chromatic_interval),
 pitches (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell
 attribute), 1053 pitchtools.calculate_melodic_chromatic_interval_class()
 pitches (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn
 attribute), 1055 (in module ab-
 pitches (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow
 attribute), 1057 pitchtools.calculate_melodic_counterpoint_interval()
 pitches_by_row (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray
 attribute), 1050 (in module ab-
 pitchtools.Accidental (class in ab- 1271
 jad.tools.pitchtools.Accidental.Accidental), pitchtools.calculate_melodic_counterpoint_interval_class()
 1129 (in module ab-
 pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs() jad.tools.pitchtools.calculate_melodic_counterpoint_interval_class()
 (in module ab- 1271
 jad.tools.pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs() pitchtools.calculate_melodic_diatonic_interval()
 1268 (in module ab-
 pitchtools.all_are_named_chromatic_pitch_tokens() jad.tools.pitchtools.calculate_melodic_diatonic_interval),
 (in module ab- 1272
 jad.tools.pitchtools.all_are_named_chromatic_pitch_tokens() pitchtools.calculate_melodic_diatonic_interval_class()
 1269 (in module ab-
 pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_abbreviation() jad.tools.pitchtools.calculate_melodic_diatonic_interval_class),
 (in module ab- 1272
 jad.tools.pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_abbreviation() pitchtools.chromatic_pitch_class_name_to_chromatic_p
 1269 (in module ab-
 pitchtools.apply_accidental_to_named_chromatic_pitch() jad.tools.pitchtools.chromatic_pitch_class_name_to_chromatic_p
 (in module ab- 1272
 jad.tools.pitchtools.apply_accidental_to_named_chromatic_pitch() pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name()
 1269 (in module ab-
 pitchtools.calculate_harmonic_chromatic_interval() jad.tools.pitchtools.chromatic_pitch_class_name_to_diatonic_pit
 (in module ab- 1272
 jad.tools.pitchtools.calculate_harmonic_chromatic_interval() pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name()
 1269 (in module ab-
 pitchtools.calculate_harmonic_chromatic_interval_class() jad.tools.pitchtools.chromatic_pitch_class_number_to_chromatic,
 (in module ab- 1272
 jad.tools.pitchtools.calculate_harmonic_chromatic_interval_class() pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name,
 1270 (in module ab-
 pitchtools.calculate_harmonic_counterpoint_interval() jad.tools.pitchtools.chromatic_pitch_class_number_to_chromatic,
 (in module ab- 1273
 jad.tools.pitchtools.calculate_harmonic_counterpoint_interval() pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name,
 1270 (in module ab-
 pitchtools.calculate_harmonic_counterpoint_interval_class() jad.tools.pitchtools.chromatic_pitch_class_number_to_chromatic,
 (in module ab- 1273
 jad.tools.pitchtools.calculate_harmonic_counterpoint_interval_class() pitchtools.chromatic_pitch_class_number_to_diatonic_pitch_class_number
 1270 (in module ab-
 pitchtools.calculate_harmonic_diatonic_interval() jad.tools.pitchtools.chromatic_pitch_class_number_to_diatonic_p

1274 (in module ab-
pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_name() jad.tools.pitchtools.chromatic_pitch_number_to_diatonic_pitch_c
(in module ab- 1277
jad.tools.pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_name() jad.tools.pitchtools.chromatic_pitch_number_to_diatonic_pitch_number()
1274 (in module ab-
pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_number() jad.tools.pitchtools.chromatic_pitch_number_to_diatonic_pitch_r
(in module ab- 1277
jad.tools.pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_number() jad.tools.pitchtools.chromatic_pitch_number_to_octave_number()
1274 (in module ab-
pitchtools.chromatic_pitch_name_to_chromatic_pitch_number() jad.tools.pitchtools.chromatic_pitch_number_to_octave_number()
(in module ab- 1277
jad.tools.pitchtools.chromatic_pitch_name_to_chromatic_pitch_number() jad.tools.pitchtools.ChromaticIntervalClassObject (class in ab-
1274 jad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticInter
pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_name() 1064
(in module ab- pitchtools.ChromaticIntervalObject (class in ab-
jad.tools.pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_name() jad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalO
1275 1066
pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_number() jad.tools.pitchtools.ChromaticObject (class in ab-
(in module ab- jad.tools.pitchtools.ChromaticObject.ChromaticObject),
jad.tools.pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_number(),
1275 pitchtools.ChromaticPitchObject (class in ab-
pitchtools.chromatic_pitch_name_to_diatonic_pitch_name() jad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject)
(in module ab- 1069
jad.tools.pitchtools.chromatic_pitch_name_to_diatonic_pitch_name() jad.tools.pitchtools.clef_and_staff_position_number_to_named_chromatic_pitch()
1275 (in module ab-
pitchtools.chromatic_pitch_name_to_diatonic_pitch_number() jad.tools.pitchtools.clef_and_staff_position_number_to_named_c
(in module ab- 1277
jad.tools.pitchtools.chromatic_pitch_name_to_diatonic_pitch_number() jad.tools.pitchtools.contains_subsegment() (in module ab-
1275 jad.tools.pitchtools.contains_subsegment),
pitchtools.chromatic_pitch_name_to_octave_number() 1278
(in module ab- pitchtools.CounterpointIntervalClassObject (class in ab-
jad.tools.pitchtools.chromatic_pitch_name_to_octave_number() jad.tools.pitchtools.CounterpointIntervalClassObject.Counterpoint
1275 1070
pitchtools.chromatic_pitch_names_string_to_named_chromatic_pitch_list() jad.tools.pitchtools.CounterpointIntervalObject (class in ab-
(in module ab- jad.tools.pitchtools.CounterpointIntervalObject.CounterpointInter
jad.tools.pitchtools.chromatic_pitch_names_string_to_named_chromatic_pitch_list(),
1276 pitchtools.CounterpointObject (class in ab-
pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number() jad.tools.pitchtools.CounterpointObject.CounterpointObject),
(in module ab- 1073
jad.tools.pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number() jad.tools.pitchtools.diatonic_pitch_class_name_to_chromatic_pit
1276 (in module ab-
pitchtools.chromatic_pitch_number_to_chromatic_pitch_class_number() jad.tools.pitchtools.diatonic_pitch_class_name_to_chromatic_pit
(in module ab- 1278
jad.tools.pitchtools.chromatic_pitch_number_to_chromatic_pitch_class_number() jad.tools.pitchtools.diatonic_pitch_class_name_to_diatonic_pitch
1276 (in module ab-
pitchtools.chromatic_pitch_number_to_chromatic_pitch_name() jad.tools.pitchtools.diatonic_pitch_class_name_to_diatonic_pitch
(in module ab- 1278
jad.tools.pitchtools.chromatic_pitch_number_to_chromatic_pitch_class_number() jad.tools.pitchtools.diatonic_pitch_class_name_to_chromatic_pitch
1276 (in module ab-
pitchtools.chromatic_pitch_number_to_chromatic_pitch_triple() jad.tools.pitchtools.diatonic_pitch_class_name_to_chromatic_p
(in module ab- 1279
jad.tools.pitchtools.chromatic_pitch_number_to_chromatic_pitch_triple() jad.tools.pitchtools.diatonic_pitch_class_name_to_diatonic_pitch_class_name()
1277 (in module ab-
pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_number() jad.tools.pitchtools.diatonic_pitch_class_name_to_diatonic_pit


```

1279 pitchtools.DiatonicPitchClassObject (class in ab-
pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_name() jad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClass
(in module ab- 1079
jad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_name() pitchtools.DiatonicPitchObject (class in ab-
1279 jad.tools.pitchtools.DiatonicPitchObject.DiatonicPitchObject),
pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_number() 1081
(in module ab- pitchtools.expr_has_duplicate_named_chromatic_pitch()
jad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_number(), module ab-
1279 jad.tools.pitchtools.expr_has_duplicate_named_chromatic_pitch()),
pitchtools.diatonic_pitch_name_to_chromatic_pitch_name() 1281
(in module ab- pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class()
jad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_name(), module ab-
1279 jad.tools.pitchtools.expr_has_duplicate_numbered_chromatic_pitch()),
pitchtools.diatonic_pitch_name_to_chromatic_pitch_number() 1281
(in module ab- pitchtools.expr_to_melodic_chromatic_interval_segment()
jad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_number(), module ab-
1280 jad.tools.pitchtools.expr_to_melodic_chromatic_interval_segment()),
pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_name() 1282
(in module ab- pitchtools.get_named_chromatic_pitch_from_pitch_carrier()
jad.tools.pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_name(), module ab-
1280 jad.tools.pitchtools.get_named_chromatic_pitch_from_pitch_carrier()),
pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_number() 1282
(in module ab- pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier()
jad.tools.pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_number(), module ab-
1280 jad.tools.pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier()),
pitchtools.diatonic_pitch_name_to_diatonic_pitch_number() 1282
(in module ab- pitchtools.harmonic_chromatic_interval_class_number_dictionary()
jad.tools.pitchtools.diatonic_pitch_name_to_diatonic_pitch_number(), module ab-
1280 jad.tools.pitchtools.harmonic_chromatic_interval_class_number_dictionary()),
pitchtools.diatonic_pitch_number_to_chromatic_pitch_number() 1283
(in module ab- pitchtools.HarmonicChromaticInterval (class in ab-
jad.tools.pitchtools.diatonic_pitch_number_to_chromatic_pitch_number() pitchtools.HarmonicChromaticInterval.HarmonicChrom
1280 pitchtools.HarmonicChromaticIntervalClass (class in ab-
pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_name() jad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicC
(in module ab- jad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicC
jad.tools.pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_name()),
1281 pitchtools.HarmonicChromaticIntervalClassVector
pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_number() (class in ab-
(in module ab- jad.tools.pitchtools.HarmonicChromaticIntervalClassVector.Harmon
jad.tools.pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_number()),
1281 pitchtools.HarmonicChromaticIntervalSegment
pitchtools.diatonic_pitch_number_to_diatonic_pitch_name() (class in ab-
(in module ab- jad.tools.pitchtools.HarmonicChromaticIntervalSegment.Harmon
jad.tools.pitchtools.diatonic_pitch_number_to_diatonic_pitch_name()),
1281 pitchtools.HarmonicChromaticIntervalSet (class in ab-
pitchtools.DiatonicIntervalClassObject (class in ab- jad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChr
jad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject),
1075 pitchtools.HarmonicCounterpointInterval (class in ab-
pitchtools.DiatonicIntervalObject (class in ab- jad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCou
jad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject),
1076 pitchtools.HarmonicCounterpointIntervalClass
pitchtools.DiatonicObject (class in ab- (class in ab-
jad.tools.pitchtools.DiatonicObject.DiatonicObject), jad.tools.pitchtools.HarmonicCounterpointIntervalClass.Harmoni
1078 1144

```

pitchtools.HarmonicDiatonicInterval (class in ab- 1285
 jad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval), version_equivalent_diatonic_interval_classes()
 1145 (in module ab-
 pitchtools.HarmonicDiatonicIntervalClass (class in ab- jad.tools.pitchtools.inventory_inversion_equivalent_diatonic_inter-
 jad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass), 1285
 1147 pitchtools.inversion_equivalent_chromatic_interval_class_number_dictionary()
 pitchtools.HarmonicDiatonicIntervalClassSet (in module ab-
 (class in ab- jad.tools.pitchtools.inversion_equivalent_chromatic_interval_class-
 jad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet), 1086
 1149 pitchtools.InversionEquivalentChromaticIntervalClass
 pitchtools.HarmonicDiatonicIntervalSegment (class in ab-
 (class in ab- jad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.In-
 jad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment), 1157
 1152 pitchtools.InversionEquivalentChromaticIntervalClassSegment
 pitchtools.HarmonicDiatonicIntervalSet (class in ab- (class in ab-
 jad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet), pitchtools.inversion_equivalent_chromatic_interval_class_set()
 1154 1159
 pitchtools.HarmonicIntervalClassObject (class in ab- pitchtools.InversionEquivalentChromaticIntervalClassSet
 jad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject), in ab-
 1082 jad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet
 pitchtools.HarmonicIntervalObject (class in ab- 1161
 jad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject), pitchtools.inversion_equivalent_chromatic_interval_class_vector()
 1084 (class in ab-
 pitchtools.HarmonicObject (class in ab- jad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVe-
 jad.tools.pitchtools.HarmonicObject.HarmonicObject), 1164
 1086 pitchtools.InversionEquivalentDiatonicIntervalClass
 pitchtools.insert_and_transpose_nested_subruns_in_chromatic_pitch_class_number_list() (class in ab-
 (in module ab- jad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.Inv-
 jad.tools.pitchtools.insert_and_transpose_nested_subruns_in_chromatic_pitch_class_number_list), 1087
 1283 pitchtools.InversionEquivalentDiatonicIntervalClassSegment
 pitchtools.instantiate_pitch_and_interval_test_collection() (class in ab-
 (in module ab- jad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSeg-
 jad.tools.pitchtools.instantiate_pitch_and_interval_test_collection), 1087
 1284 pitchtools.InversionEquivalentDiatonicIntervalClassVector
 pitchtools.IntervalClassObjectSegment (class in ab- (class in ab-
 jad.tools.pitchtools.IntervalClassObjectSegment.IntervalClassObjectSegment), pitchtools.inversion_equivalent_diatonic_interval_class_vector()
 1087 1171
 pitchtools.IntervalClassObjectSet (class in ab- pitchtools.is_alphabetic_accidental_abbreviation()
 jad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet), module ab-
 1089 jad.tools.pitchtools.is_alphabetic_accidental_abbreviation(),
 pitchtools.IntervalObject (class in ab- 1286
 jad.tools.pitchtools.IntervalObject.IntervalObject) pitchtools.is_chromatic_pitch_class_name()
 1092 (in module ab-
 pitchtools.IntervalObjectClass (class in ab- jad.tools.pitchtools.is_chromatic_pitch_class_name(),
 jad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass), 1286
 1093 pitchtools.is_chromatic_pitch_class_name_octave_number_pair()
 pitchtools.IntervalObjectSegment (class in ab- (in module ab-
 jad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment), jad.tools.pitchtools.is_chromatic_pitch_class_name_octave_num-
 1095 1287
 pitchtools.IntervalObjectSet (class in ab- pitchtools.is_chromatic_pitch_class_number()
 jad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet), (in module ab-
 1097 jad.tools.pitchtools.is_chromatic_pitch_class_number(),
 pitchtools.inventory_aggregate_subsets() (in module ab- 1287
 jad.tools.pitchtools.inventory_aggregate_subsets), pitchtools.is_chromatic_pitch_name() (in module ab-

jad.tools.pitchtools.is_chromatic_pitch_name(), 1287	pitchtools.list_chromatic_pitch_numbers_in_expr() (in module ab-
pitchtools.is_chromatic_pitch_number() (in module ab- jad.tools.pitchtools.is_chromatic_pitch_number), 1287	jad.tools.pitchtools.list_chromatic_pitch_numbers_in_expr(), 1292
pitchtools.is_diatonic_pitch_class_name() (in module ab- jad.tools.pitchtools.is_diatonic_pitch_class_name), 1287	pitchtools.list_harmonic_chromatic_intervals_in_expr() (in module ab-
pitchtools.is_diatonic_pitch_class_number() (in module ab- jad.tools.pitchtools.is_diatonic_pitch_class_number), 1288	jad.tools.pitchtools.list_harmonic_chromatic_intervals_in_expr(), 1292
pitchtools.is_diatonic_pitch_name() (in module ab- jad.tools.pitchtools.is_diatonic_pitch_name), 1288	pitchtools.list_harmonic_diatonic_intervals_in_expr() (in module ab-
pitchtools.is_diatonic_pitch_number() (in module ab- jad.tools.pitchtools.is_diatonic_pitch_number), 1288	jad.tools.pitchtools.list_harmonic_diatonic_intervals_in_expr(), 1292
pitchtools.is_diatonic_quality_abbreviation() (in module ab- jad.tools.pitchtools.is_diatonic_quality_abbreviatipitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise() (in module ab-	pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise() (in module ab-
pitchtools.is_harmonic_diatonic_interval_abbreviation() (in module ab- jad.tools.pitchtools.is_harmonic_diatonic_interval_abbreviatipitchtools.list_melodic_chromatic_interval_numbers_pairwise() (in module ab-	pitchtools.list_melodic_chromatic_interval_numbers_pairwise() (in module ab-
pitchtools.is_melodic_diatonic_interval_abbreviation() (in module ab- jad.tools.pitchtools.is_melodic_diatonic_interval_abbreviatipitchtools.list_named_chromatic_pitches_in_expr() (in module ab-	pitchtools.list_named_chromatic_pitches_in_expr() (in module ab-
pitchtools.is_named_chromatic_pitch_token() (in module ab- jad.tools.pitchtools.is_named_chromatic_pitch_tokenpitchtools.list_numbered_chromatic_pitch_classes_in_expr() (in module ab-	pitchtools.list_numbered_chromatic_pitch_classes_in_expr() (in module ab-
pitchtools.is_octave_tick_string() (in module ab- jad.tools.pitchtools.is_octave_tick_string), 1289	pitchtools.list_numbered_chromatic_pitch_classes_in_expr() (in module ab-
pitchtools.is_pitch_carrier() (in module ab- jad.tools.pitchtools.is_pitch_carrier), 1289	pitchtools.list_octave_transpositions_of_pitch_carrier_within_pitch_range() (in module ab-
pitchtools.is_pitch_class_octave_number_string() (in module ab- jad.tools.pitchtools.is_pitch_class_octave_number_string), 1290	pitchtools.list_octave_transpositions_of_pitch_carrier_w (in module ab-
pitchtools.is_symbolic_accidental_string() (in module ab- jad.tools.pitchtools.is_symbolic_accidental_string), 1290	pitchtools.list_ordered_named_chromatic_pitch_pairs_from_expr_1_to_expr_2() (in module ab-
pitchtools.is_symbolic_pitch_range_string() (in module ab- jad.tools.pitchtools.is_symbolic_pitch_range_string), 1290	pitchtools.list_ordered_named_chromatic_pitch_pairs_fr (in module ab-
pitchtools.iterate_named_chromatic_pitch_pairs_in_expr() (in module ab- jad.tools.pitchtools.iterate_named_chromatic_pitch_pairs_in_expr(), 1290	pitchtools.list_unordered_named_chromatic_pitch_pairs_in_expr() (in module ab- jad.tools.pitchtools.list_unordered_named_chromatic_pitch_pairs 1296
	pitchtools.make_n_middle_c_centered_pitches() (in module ab-
	jad.tools.pitchtools.make_n_middle_c_centered_pitches), 1297
	pitchtools.MelodicChromaticInterval (class in ab- jad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval 1174
	pitchtools.MelodicChromaticIntervalClass (class in ab- jad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass 1176
	pitchtools.MelodicChromaticIntervalClassSegment (class in ab-
	jad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment 1178

pitchtools.MelodicChromaticIntervalClassVector	jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment
(class in ab-	1209
jad.tools.pitchtools.MelodicChromaticIntervalClassVector	pitchtools.MelodicChromaticIntervalClassVector
1180	jad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet
pitchtools.MelodicChromaticIntervalSegment	1211
(class in ab-	pitchtools.NamedChromaticPitchSegment (class in ab-
jad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment	pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment
1183	1215
pitchtools.MelodicChromaticIntervalSet (class in ab-	pitchtools.NamedChromaticPitchSet (class in ab-
jad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet	pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet
1186	1218
pitchtools.MelodicCounterpointInterval (class in ab-	pitchtools.NamedChromaticPitchVector (class in ab-
jad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval	pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector
1189	1221
pitchtools.MelodicCounterpointIntervalClass (class in ab-	pitchtools.NamedDiatonicPitch (class in ab-
jad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass	pitchtools.NamedDiatonicPitch.NamedDiatonicPitch)
1191	1224
pitchtools.MelodicDiatonicInterval (class in ab-	pitchtools.NamedDiatonicPitchClass (class in ab-
jad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval	pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass
1193	1228
pitchtools.MelodicDiatonicIntervalClass (class in ab-	pitchtools.NumberedChromaticPitch (class in ab-
jad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass	pitchtools.NumberedChromaticPitch.NumberedChromaticPitch
1195	1230
pitchtools.MelodicDiatonicIntervalSegment (class in ab-	pitchtools.NumberedChromaticPitchClass (class in ab-
jad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment	pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass
1197	1232
pitchtools.MelodicDiatonicIntervalSet (class in ab-	pitchtools.NumberedChromaticPitchClassColorMap (class in ab-
jad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet	pitchtools.NumberedChromaticPitchClassColorMap.NamedChromaticPitchClassColorMap
1199	1234
pitchtools.MelodicIntervalClassObject (class in ab-	pitchtools.NamedChromaticPitchClassSegment (class in ab-
jad.tools.pitchtools.MelodicIntervalClassObject.MelodicIntervalClassObject	pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment
1100	1236
pitchtools.MelodicIntervalObject (class in ab-	pitchtools.NumberedChromaticPitchClassSet (class in ab-
jad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject	pitchtools.NumberedChromaticPitchClassSet.NamedChromaticPitchClassSet
1101	1239
pitchtools.MelodicObject (class in ab-	pitchtools.NumberedChromaticPitchClassVector (class in ab-
jad.tools.pitchtools.MelodicObject.MelodicObject)	pitchtools.NumberedChromaticPitchClassVector.NamedChromaticPitchClassVector
1103	1243
pitchtools.named_chromatic_pitch_and_clef_to_staff_position	pitchtools.named_chromatic_pitch_and_clef_to_staff_position
(in module ab-	1297
jad.tools.pitchtools.named_chromatic_pitch_and_clef_to_staff_position	pitchtools.named_chromatic_pitch_and_clef_to_staff_position
1297	1246
pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches	pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches
(in module ab-	1297
jad.tools.pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches	pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches
1297	1249
pitchtools.NamedChromaticPitch (class in ab-	pitchtools.NamedChromaticPitchClass (class in ab-
jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch	pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass
1202	1250
pitchtools.NamedChromaticPitchClass (class in ab-	pitchtools.NumberedObject (class in ab-
jad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass	pitchtools.NumberedObject.NumberedObject)
1207	1251
pitchtools.NamedChromaticPitchClassSegment (class in ab-	pitchtools.NumberedPitchClassObject (class in ab-
(class in ab-	pitchtools.NumberedPitchClassObject.NumberedPitchClassObject
1106	

pitchtools.NumberedPitchObject (class in ab- 1123
 jad.tools.pitchtools.NumberedPitchObject.NumberedPitchObjectSegment (class in ab-
 1107 jad.tools.pitchtools.PitchObjectSegment.PitchObjectSegment),
 pitchtools.ObjectSegment (class in ab- 1124
 jad.tools.pitchtools.ObjectSegment.ObjectSegment),
 pitchtools.ObjectSet (class in ab- 1126
 jad.tools.pitchtools.ObjectSet.ObjectSet),
 pitchtools.PitchObjectSet (class in ab-
 1109 jad.tools.pitchtools.PitchObjectSet.PitchObjectSet),
 pitchtools.PitchRange (class in ab- 1126
 jad.tools.pitchtools.PitchRange.PitchRange),
 pitchtools.ObjectVector (class in ab- 1260
 jad.tools.pitchtools.ObjectVector.ObjectVector),
 pitchtools.PitchRangeInventory (class in ab-
 1114 jad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory),
 pitchtools.octave_number_to_octave_tick_string() 1262
 (in module ab- pitchtools.register_chromatic_pitch_class_numbers_by_chromatic_pitch_nu
 jad.tools.pitchtools.octave_number_to_octave_tick_string), (in module ab-
 1298 jad.tools.pitchtools.register_chromatic_pitch_class_numbers_by_
 pitchtools.octave_tick_string_to_octave_number() 1300
 (in module ab- pitchtools.respell_named_chromatic_pitches_in_expr_with_flats()
 jad.tools.pitchtools.octave_tick_string_to_octave_number), (in module ab-
 1298 jad.tools.pitchtools.respell_named_chromatic_pitches_in_expr_w
 pitchtools.OctaveTranspositionMapping (class in ab- 1300
 jad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping),
 pitchtools.OctaveTranspositionMappingComponent 1300
 (class in ab- jad.tools.pitchtools.respell_named_chromatic_pitches_in_expr_w
 jad.tools.pitchtools.OctaveTranspositionMappingComponent),
 1254 (in module ab- pitchtools.set_ascending_named_chromatic_pitches_on_
 pitchtools.OctaveTranspositionMappingInventory 1301
 (class in ab- jad.tools.pitchtools.set_ascending_named_chromatic_pitches_on_tie_chains_in_expr()
 jad.tools.pitchtools.OctaveTranspositionMappingInventory),
 1256 (in module ab- pitchtools.set_ascending_named_diatonic_pitches_on_tie_
 pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number() 1301
 (in module ab- pitchtools.set_default_accidental_spelling()
 jad.tools.pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number),
 1298 (in module ab- pitchtools.set_default_accidental_spelling),
 pitchtools.permute_named_chromatic_pitch_carrier_list_by_twelve_tone_towers() 1302
 (in module ab- pitchtools.set_default_accidental_spelling),
 jad.tools.pitchtools.permute_named_chromatic_pitch_carrier_list_by_twelve_tone_towers()
 1299 (in module ab- pitchtools.set_default_accidental_spelling),
 pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name() 1303
 (in module ab- pitchtools.set_default_accidental_spelling),
 jad.tools.pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name()
 1299 (in module ab- pitchtools.set_default_accidental_spelling),
 pitchtools.PitchClassObject (class in ab- 1303
 jad.tools.pitchtools.PitchClassObject.PitchClassObject),
 1116 pitchtools.spell_chromatic_interval_number()
 pitchtools.PitchClassObjectSegment (class in ab- (in module ab-
 jad.tools.pitchtools.PitchClassObjectSegment.PitchClassObjectSegment),
 1118 jad.tools.pitchtools.spell_chromatic_interval_number),
 pitchtools.PitchClassObjectSet (class in ab- 1303
 jad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet),
 1120 (in module ab- pitchtools.spell_chromatic_pitch_number()
 pitchtools.PitchObject (class in ab- 1304
 jad.tools.pitchtools.PitchObject.PitchObject),
 pitchtools.split_chromatic_pitch_class_name()

(in module ab- pop() (abjad.tools.containertools.Cluster.Cluster.Cluster method), 339
 jad.tools.pitchtools.split_chromatic_pitch_class_name), method), 339
 1304 pop() (abjad.tools.containertools.Container.Container.Container method), 346
 pitchtools.suggest_clef_for_named_chromatic_pitches() method), 346
 (in module ab- pop() (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer method), 353
 jad.tools.pitchtools.suggest_clef_for_named_chromatic_pitches() method), 353
 1304 pop() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 381
 pitchtools.symbolic_accidental_string_to_alphabetic_accidental_abbreviation() method), 381
 (in module ab- pop() (abjad.tools.contexttools.Context.Context.Context method), 389
 jad.tools.pitchtools.symbolic_accidental_string_to_alphabetic_accidental_abbreviation(), method), 389
 1304 pop() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 415
 pitchtools.transpose_chromatic_pitch_by_melodic_chromatic_interval_segment() method), 415
 (in module ab- pop() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary method), 490
 jad.tools.pitchtools.transpose_chromatic_pitch_by_melodic_chromatic_interval_segment(), method), 490
 1305 pop() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 506
 pitchtools.transpose_chromatic_pitch_class_number_to_chromatic_pitch_class_number_neighbor() method), 506
 (in module ab- pop() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 594
 jad.tools.pitchtools.transpose_chromatic_pitch_class_number_to_chromatic_pitch_class_number_neighbor(), method), 594
 1305 pop() (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer method), 580
 pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping() method), 580
 (in module ab- pop() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 659
 jad.tools.pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping(), method), 659
 1305 pop() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 864
 pitchtools.transpose_named_chromatic_pitch_by_melodic_chromatic_interval_and_respell() method), 864
 (in module ab- pop() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 863
 jad.tools.pitchtools.transpose_named_chromatic_pitch_by_melodic_chromatic_interval_and_respell(), method), 863
 1306 pop() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 876
 pitchtools.transpose_pitch_carrier_by_melodic_interval() method), 876
 (in module ab- pop() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock method), 859
 jad.tools.pitchtools.transpose_pitch_carrier_by_melodic_interval() method), 859
 1307 pop() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 884
 pitchtools.transpose_pitch_expr_into_pitch_range() method), 884
 (in module ab- pop() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 938
 jad.tools.pitchtools.transpose_pitch_expr_into_pitch_range() method), 938
 1307 pop() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 986
 pitchtools.TwelveToneRow (class in ab- method), 986
 jad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 997
 1265 pop() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 997
 poll() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1970
 method), 1970 pop() (abjad.tools.measuretools.Measure.Measure.Measure method), 1007
 method), 1007 pop() (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner method), 1058
 method), 1058 pop() (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 1135
 method), 1135 pop() (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 1165
 method), 1165 pop() (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 1172
 method), 1172 pop() (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 1181
 method), 1181 pop() (abjad.tools.chordtools.Chord.Chord.Chord method), 1222
 method), 1222 pop() (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1896
 method), 1896 pop() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector method), 1244
 method), 1244

pop() (abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector method), 1115

pop() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1252

pop() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1257

pop() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1263

pop() (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer method), 1332

pop() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1374

pop() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1383

pop() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1391

pop() (abjad.tools.scoretools.Score.Score.Score method), 1400

pop() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1409

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1418

pop() (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner method), 1516

pop() (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner method), 1523

pop() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1532

pop() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1540

pop() (abjad.tools.spannertoolsDirectedSpanner.DirectedSpanner.DirectedSpanner method), 1504

pop() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1547

pop() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1553

pop() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1562

pop() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1568

pop() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1574

pop() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method), 1581

pop() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner method), 1589

pop() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1595

pop() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner method), 1602

pop() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method), 1608

pop() (abjad.tools.spannertools.Spanner.Spanner.Spanner method), 1510

pop() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner method), 1615

pop() (abjad.tools.spannertools.TablatureSpanner.TablatureSpanner.TablatureSpanner method), 1622

pop() (abjad.tools.spannertools.TempoSpanner.TempoSpanner.TempoSpanner method), 1628

pop() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method), 1635

pop() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1659

pop() (abjad.tools.stafftools.Staff.Staff.Staff method), 1668

pop() (abjad.tools.stafftools.TieSpanner.TieSpanner.TieSpanner method), 1688

pop() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1708

pop() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1737

pop() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1836

pop() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1845

pop() (abjad.tools.voicetools.Voice.Voice.Voice method), 1871

pop() (abjad.tools.pitchtools.PitchArray.PitchArray.PitchArray method), 1051

pop() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 228

pop() (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 236

pop() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 244

pop() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 253

pop() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 260

pop() (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner method), 1517

pop() (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner method), 1523

pop() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1532

pop() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1541

pop() (abjad.tools.spannertools.DirectedSpanner.DirectedSpanner.DirectedSpanner method), 1504

pop() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1548

pop() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1554

pop() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1562

pop() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1568

pop_left() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.ChordQualityIndicator.ChordQualityIndicator (method), 1575
 pop_left() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner (method), 1582
 pop_left() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner (method), 1589
 pop_left() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner (method), 1596
 pop_left() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner (method), 1602
 pop_left() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner (method), 1609
 pop_left() (abjad.tools.spannertools.Spanner.Spanner.Spanner (method), 1510
 pop_left() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner (method), 1616
 pop_left() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner (method), 1622
 pop_left() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner (method), 1629
 pop_left() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner (method), 1636
 pop_left() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner (method), 1689
 pop_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray (method), 1051
 popitem() (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig (method), 1896
 popitem() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary (method), 1903
 popitem() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph (method), 1967
 popitem() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector (method), 1135
 popitem() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector (method), 1165
 popitem() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector (method), 1172
 popitem() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector (method), 1181
 popitem() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector (method), 1222
 popitem() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector (method), 1245
 popitem() (abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector (method), 1115
 popitem() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval (method), 1708
 popitem() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary (method), 1737
 position (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree (attribute), 1425
 position (abjad.tools.sequencetools.Tree.Tree.Tree (attribute), 1438

attribute), 981

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure.HairpinSpanner.HairpinSpanner.HairpinSpanner.
 attribute), 992 attribute), 1558

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.measuretools.Measure.Measure.Measure jad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.H
 attribute), 1003 attribute), 1565

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic.jad.tools.spannertools.HorizontalBracketSpanner.HorizontalBrack
 attribute), 1033 attribute), 1571

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.notetools.Note.Note.Note attribute), jad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.Me
 1038 attribute), 1578

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest.jad.tools.spannertools.OctavationSpanner.OctavationSpanner.Octa
 attribute), 1309 attribute), 1585

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.resttools.Rest.Rest.Rest attribute), jad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.
 1311 attribute), 1592

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff jad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.Pia
 attribute), 1370 attribute), 1598

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff jad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner
 attribute), 1387 attribute), 1605

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.scoretools.Score.Score.Score at- jad.tools.spannertools.Spanner.Spanner.Spanner
 tribute), 1396 attribute), 1507

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup jad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.Staff
 attribute), 1405 attribute), 1612

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.skiptools.Skip.Skip.Skip attribute), jad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextS
 1496 attribute), 1619

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner.jad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner
 attribute), 1513 attribute), 1625

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner.TrillSpanner
 attribute), 1520 attribute), 1632

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner.jad.tools.spannertools.RhythmicStaff.RhythmicStaff.RhythmicStaff
 attribute), 1528 attribute), 1655

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner.jad.tools.spannertools.StaffSpanner.StaffSpanner.StaffSpanner
 attribute), 1536 attribute), 1664

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.spannertoolsDirectedSpannerDirectedSpannerDirectedSpanner.jad.tools.spannertools.TieChain.TieChain.TieChain
 attribute), 1501 attribute), 1683

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner.jad.tools.spannertools.TieSpanner.TieSpanner.TieSpanner
 attribute), 1544 attribute), 1685

preprolated_duration (ab- preprolated_duration (ab-
 jad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner.jad.tools.spannertools.FixedDurationTuplet.FixedDurationTuplet.Fi

attribute), 1831
 preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1841
 preprolated_duration (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1867
 pretty_rtm_format (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer attribute), 1327
 pretty_rtm_format (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1340
 pretty_rtm_format (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode attribute), 1320
 prev (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell attribute), 1053
 prev_vertical_moment (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment attribute), 1860
 primary_clefs (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 489
 primary_clefs (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 495
 primary_clefs (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 501
 primary_clefs (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 507
 primary_clefs (abjad.tools.instrumenttools.BartoneSaxophone.BartoneSaxophone.BartoneSaxophone attribute), 519
 primary_clefs (abjad.tools.instrumenttools.BartoneVoice.BartoneVoice.BartoneVoice attribute), 525
 primary_clefs (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 531
 primary_clefs (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 537
 primary_clefs (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 561
 primary_clefs (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 543
 primary_clefs (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 549
 primary_clefs (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 555
 primary_clefs (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 513
 primary_clefs (abjad.tools.instrumenttools.Cello.Cello.Cello attribute), 567
 primary_clefs (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute), 573
 primary_clefs (abjad.tools.instrumenttools.Contrabass.Contrabass attribute), 579
 primary_clefs (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet attribute), 585
 primary_clefs (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute attribute), 591
 primary_clefs (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon attribute), 603
 primary_clefs (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone attribute), 597
 primary_clefs (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice attribute), 609
 primary_clefs (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet attribute), 617
 primary_clefs (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn attribute), 617
 primary_clefs (abjad.tools.instrumenttools.Flute.Flute.Flute attribute), 617
 primary_clefs (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn attribute), 617
 primary_clefs (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel attribute), 638
 primary_clefs (abjad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 644
 primary_clefs (abjad.tools.instrumenttools.Harp.Harp.Harp attribute), 650
 primary_clefs (abjad.tools.instrumenttools.Harpsichord.Harpsichord attribute), 656
 primary_clefs (abjad.tools.instrumenttools.Marimba.Marimba attribute), 665
 primary_clefs (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice attribute), 671
 primary_clefs (abjad.tools.instrumenttools.Oboe.Oboe attribute), 677
 primary_clefs (abjad.tools.instrumenttools.Piano.Piano attribute), 683
 primary_clefs (abjad.tools.instrumenttools.Piccolo.Piccolo attribute), 689
 primary_clefs (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone attribute), 695
 primary_clefs (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone attribute), 701
 primary_clefs (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice attribute), 707
 primary_clefs (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone attribute), 713
 primary_clefs (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone attribute), 719
 primary_clefs (abjad.tools.instrumenttools.TenorVoice.TenorVoice attribute), 725
 primary_clefs (abjad.tools.instrumenttools.Trumpet.Trumpet attribute), 731
 primary_clefs (abjad.tools.instrumenttools.Tuba.Tuba attribute), 737
 primary_clefs (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion attribute), 743
 primary_clefs (abjad.tools.instrumenttools.Vibraphone.Vibraphone attribute), 749
 primary_clefs (abjad.tools.instrumenttools.Viola.Viola attribute), 755

primary_clefs (abjad.tools.instrumenttools.Violin.Violin.ViolinProgramName (abjad.tools.developerscripttools.CleanScript.CleanScript.CleanScript attribute), 761

primary_clefs (abjad.tools.instrumenttools.Xylophone.Xylophone.XylophoneProgramName (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScript attribute), 767

prime_form (abjad.tools.pitchtools.NumberedChromaticPitchClassSetName (abjad.tools.pitchtools.NumberedChromaticPitchClassSetName attribute), 1240

process_args() (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.AbjadBookScript attribute), 1887

process_args() (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript attribute), 1911

process_args() (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript attribute), 1913

process_args() (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript attribute), 1918

process_args() (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript attribute), 1930

process_args() (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript attribute), 1921

process_args() (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript attribute), 1932

process_args() (abjad.tools.developerscripttools.CleanScript.CleanScript.CleanScript attribute), 1923

process_args() (abjad.tools.developerscripttools.CleanScript.CleanScript.CleanScript attribute), 1934

process_args() (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScript attribute), 1926

process_args() (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScript attribute), 1937

process_args() (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScript attribute), 1928

process_args() (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScript attribute), 1940

process_args() (abjad.tools.developerscripttools.DeveloperScript.DeveloperScript.DeveloperScript attribute), 1911

process_args() (abjad.tools.developerscripttools.DeveloperScript.DeveloperScript.DeveloperScript attribute), 1942

process_args() (abjad.tools.developerscripttools.DirectoryScript.DirectoryScript.DirectoryScript attribute), 1913

process_args() (abjad.tools.developerscripttools.DirectoryScript.DirectoryScript.DirectoryScript attribute), 1945

process_args() (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript attribute), 1930

process_args() (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript attribute), 1947

process_args() (abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript attribute), 1932

process_args() (abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript attribute), 1949

process_args() (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript.RenameModulesScript attribute), 1935

process_args() (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript.RenameModulesScript attribute), 1952

process_args() (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript attribute), 1937

process_args() (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript attribute), 225

process_args() (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript.ReplacePromptsScript attribute), 1940

process_args() (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript.ReplacePromptsScript attribute), 231

process_args() (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScript attribute), 1942

process_args() (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScript attribute), 239

process_args() (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript attribute), 1945

process_args() (abjad.tools.developerscripttools.SvnAddAllScript.SvnAddAllScript.SvnAddAllScript attribute), 248

process_args() (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript attribute), 1947

process_args() (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript attribute), 256

process_args() (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript.SvnMessageScript attribute), 1949

process_args() (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript.SvnMessageScript attribute), 267

process_args() (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript attribute), 1952

process_args() (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript attribute), 276

processed_results (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock attribute), 1888

processed_results (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock attribute), 336

program_name (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.AbjadBookScript attribute), 1887

program_name (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.AbjadBookScript attribute), 343

program_name (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript attribute), 1916

program_name (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript attribute), 350

program_name (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript attribute), 1918

program_name (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript attribute), 385

program_name (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript attribute), 1920

program_name (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript attribute), 476

prolated_duration (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 829
 prolated_duration (abjad.tools.measuretools.AnonymousMeasure attribute), 981
 prolated_duration (abjad.tools.measuretools.DynamicMeasure attribute), 992
 prolated_duration (abjad.tools.measuretools.Measure.Measure attribute), 1003
 prolated_duration (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic attribute), 1033
 prolated_duration (abjad.tools.notetools.Note.Note.Note attribute), 1038
 prolated_duration (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest attribute), 1309
 prolated_duration (abjad.tools.resttools.Rest.Rest.Rest attribute), 1312
 prolated_duration (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer attribute), 1327
 prolated_duration (abjad.tools.rhythmtreetools.RhythmTreeProfile.RhythmTreeProfile attribute), 1340
 prolated_duration (abjad.tools.rhythmtreetools.RhythmTreeProfile.RhythmTreeProfile attribute), 1320
 prolated_duration (abjad.tools.scoretools.GrandStaff.GrandStaff attribute), 1370
 prolated_duration (abjad.tools.scoretools.PianoStaff.PianoStaff attribute), 1387
 prolated_duration (abjad.tools.scoretools.Score.Score attribute), 1396
 prolated_duration (abjad.tools.scoretools.StaffGroup.StaffGroup attribute), 1405
 prolated_duration (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1496
 prolated_duration (abjad.tools.spannertools.BracketSpanner.BracketSpanner attribute), 1513
 prolated_duration (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner attribute), 1520
 prolated_duration (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner attribute), 1528
 prolated_duration (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner attribute), 1536
 prolated_duration (abjad.tools.spannertools.DirectedSpanner.DirectedSpanner attribute), 1501
 prolated_duration (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner attribute), 1544
 prolated_duration (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner attribute), 1550
 prolated_duration (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner attribute), 1558
 prolated_duration (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner attribute), 1565
 prolated_duration (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner attribute), 1571
 prolated_duration (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner attribute), 1578
 prolated_duration (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner attribute), 1585
 prolated_duration (abjad.tools.spannertools.PhrasingSpanner.PhrasingSpanner attribute), 1592
 prolated_duration (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner attribute), 1598
 prolated_duration (abjad.tools.spannertools.SlurSpanner.SlurSpanner attribute), 1605
 prolated_duration (abjad.tools.spannertools.Spanner.Spanner attribute), 1507
 prolated_duration (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner attribute), 1612
 prolated_duration (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner attribute), 1619
 prolated_duration (abjad.tools.spannertools.TextSpanner.TextSpanner attribute), 1625
 prolated_duration (abjad.tools.spannertools.TrillSpanner.TrillSpanner attribute), 1632
 prolated_duration (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff attribute), 1656
 prolated_duration (abjad.tools.stafftools.Staff.Staff attribute), 1665
 prolated_duration (abjad.tools.tietools.TieChain.TieChain attribute), 1683
 prolated_duration (abjad.tools.tietools.TieSpanner.TieSpanner attribute), 1685
 prolated_duration (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1831
 prolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet attribute), 1841
 prolated_duration (abjad.tools.voicetools.Voice.Voice attribute), 1867
 prolated_duration (abjad.tools.verticalmomenttools.VerticalMoment.VerticalMoment attribute), 1860
 prolated_duration (abjad.tools.verticalmomenttools.VerticalMoment.VerticalMoment attribute), 267
 prolated_duration (abjad.tools.componenttools.Component.Component attribute), 276
 prolated_duration (abjad.tools.clustertools.Cluster.Cluster attribute), 336
 prolated_duration (abjad.tools.containertools.Container.Container attribute), 343
 prolated_duration (abjad.tools.fixeddurationtools.FixedDurationContainer.FixedDurationContainer attribute), 350
 prolated_duration (abjad.tools.contexttools.Context.Context attribute), 385
 prolated_duration (abjad.tools.gracecontainertools.GraceContainer.GraceContainer attribute), 476
 prolated_duration (abjad.tools.hiddenstaffspanner.HiddenStaffSpanner attribute), 829
 prolated_duration (abjad.tools.horizontalbracketspanner.HorizontalBracketSpanner attribute), 981
 prolated_duration (abjad.tools.metricgridspanner.MetricGridSpanner attribute), 992

prolation (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1003
 quality_string (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject attribute), 1075
 prolation (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 1033
 quality_string (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject attribute), 1077
 prolation (abjad.tools.notetools.Note.Note.Note attribute), 1038
 quality_string (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval attribute), 1146
 prolation (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1309
 quality_string (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass attribute), 1147
 prolation (abjad.tools.resttools.Rest.Rest.Rest attribute), 1312
 quality_string (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass attribute), 1167
 prolation (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1370
 quality_string (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval attribute), 1194
 prolation (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1387
 quality_string (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass attribute), 1195
 prolation (abjad.tools.scoretools.Score.Score.Score attribute), 1396
 quality_string (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator attribute), 1808
 prolation (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1405
 quality_string (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.QualityIndicator attribute), 1814
 prolation (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1496
 quantize_to_rational() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval attribute), 1708
 prolation (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1656
 quantize_to_rational() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin attribute), 1702
 prolation (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1665
 quantize_to_rational() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin attribute), 1702
 prolation (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1831
 quantize_to_rational() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin attribute), 1705
 prolation (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1841
 quantize_to_rational() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree attribute), 1717
 prolation (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1867
 quantize_to_rational() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree attribute), 1717
 proper_parentage (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer attribute), 1327
 quantize_to_rational() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1717
 proper_parentage (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1341
 quarters_per_minute (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1341
 proper_parentage (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode attribute), 1320
 tempo_mark (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode attribute), 411
 proper_parentage (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1425
 ratio_string (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1831
 proper_parentage (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1438
 ratio_string (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1831
 proportional_notation_duration (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication attribute), 824
 rcs (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression attribute), 1492
 push_signature() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition attribute), 2041
 read() (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock attribute), 1889
 read() (abjad.tools.documentationtools.Pipe.Pipe.Pipe attribute), 1971
 read_wait() (abjad.tools.documentationtools.Pipe.Pipe.Pipe attribute), 1971
 quality (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction attribute), 1822
 quality_indicator (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass attribute), 1804
 readonly_properties (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter attribute), 1959
 quality_pair (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass attribute), 1804
 readonly_properties (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter attribute), 1959

Q

method), 2090

report() (abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck.MisfilledMeasureCheck method), 2091

report() (abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck.MispitchedTieCheck method), 2093

report() (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck method), 2094

report() (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.MissingParentCheck method), 2096

report() (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck method), 2097

report() (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck method), 2099

report() (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck method), 2100

report() (abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.OverlappingOctavationCheck method), 2102

report() (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck method), 2103

representative_boolean_train (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression attribute), 1492

representative_congruent_bases (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression attribute), 1492

residue (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass attribute), 1491

resttools.all_are_rests() (in module abjad.tools.resttools.all_are_rests), 1314

resttools.is_lilypond_rest_string() (in module abjad.tools.resttools.is_lilypond_rest_string), 1314

resttools.make_multi_measure_rests() (in module abjad.tools.resttools.make_multi_measure_rests), 1314

resttools.make_repeated_rests_from_time_signature() (in module abjad.tools.resttools.make_repeated_rests_from_time_signature), 1315

resttools.make_repeated_rests_from_time_signatures() (in module abjad.tools.resttools.make_repeated_rests_from_time_signatures), 1315

resttools.make_rests() (in module abjad.tools.resttools.make_rests), 1315

resttools.make_tied_rest() (in module abjad.tools.resttools.make_tied_rest), 1315

resttools.MultiMeasureRest (class in abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest), 1308

resttools.replace_leaves_in_expr_with_rests() (in module abjad.tools.resttools.replace_leaves_in_expr_with_rests), 1316

resttools.Rest (class in abjad.tools.resttools.Rest.Rest), 1316

resttools.set_vertical_positioning_pitch_on_rest() (abjad.tools.resttools.set_vertical_positioning_pitch_on_rest), 1316

resttools.yield_groups_of_rests_in_sequence() (abjad.tools.resttools.yield_groups_of_rests_in_sequence), 1316

retrograde() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment), 1316

retrograde() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment), 1316

retrograde() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow), 1316

retrograde() (abjad.tools.tonalitytools.Scale.Scale.Scale), 1316

reverse() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory), 381

reverse() (abjad.tools.datastructuretools.Digraph.Digraph.Digraph), 415

reverse() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory), 1906

reverse() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory), 659

reverse() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock), 864

reverse() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock), 867

reverse() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile), 876

reverse() (abjad.tools.lilypondfiletools.NonattributedBlock.NonattributedBlock.NonattributedBlock), 859

reverse() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock), 884

reverse() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory), 938

reverse() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping), 1252

reverse() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory), 1257

reverse() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory), 1263

reverse() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory), 1383

reverse() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList), 1418

reverse() (abjad.tools.timetokentools.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker), 1759

reverse() (abjad.tools.timetokentools.EqualDivisionTimeTokenMaker.EqualDivisionTimeTokenMaker.EqualDivisionTimeTokenMaker), 1769

reverse()	(abjad.tools.timetokentools.IncisedTimeTokenMaker.IncisedTimeTokenMaker.IncisedTimeTokenMaker	reverse()	(abjad.tools.timetokentools.TimeTokenMaker.TimeTokenMaker.TimeTokenMaker
method), 1760	method), 1666	method), 1666	method), 1666
reverse()	(abjad.tools.timetokentools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker
method), 1772	method), 1328	method), 1328	method), 1328
reverse()	(abjad.tools.timetokentools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker
method), 1775	method), 1341	method), 1341	method), 1341
reverse()	(abjad.tools.timetokentools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker
method), 1778	method), 1320	method), 1320	method), 1320
reverse()	(abjad.tools.timetokentools.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker
method), 1781	method), 1901	method), 1901	method), 1901
reverse()	(abjad.tools.timetokentools.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker	reverse()	(abjad.tools.timetokentools.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker
method), 1762	method), 1955	method), 1955	method), 1955
reverse()	(abjad.tools.timetokentools.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker
method), 1784	method), 1957	method), 1957	method), 1957
reverse()	(abjad.tools.timetokentools.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker
method), 1787	method), 1957	method), 1957	method), 1957
reverse()	(abjad.tools.timetokentools.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker.SkipFilledTimeTokenMaker
method), 1789	method), 1962	method), 1962	method), 1962
reverse()	(abjad.tools.timetokentools.TimeTokenMaker.TimeTokenMaker.TimeTokenMaker	reverse()	(abjad.tools.timetokentools.TimeTokenMaker.TimeTokenMaker.TimeTokenMaker
method), 1764	method), 1964	method), 1964	method), 1964
reverse()	(abjad.tools.timetokentools.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker
method), 1793	method), 1965	method), 1965	method), 1965
reverse()	(abjad.tools.timetokentools.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker
method), 1796	method), 1962	method), 1962	method), 1962
reverse()	(abjad.tools.timetokentools.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker	reverse()	(abjad.tools.timetokentools.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker
method), 1799	method), 1964	method), 1964	method), 1964
reverse()	(abjad.tools.timetokentools.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker	reverse()	(abjad.tools.timetokentools.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker
method), 1766	method), 1966	method), 1966	method), 1966
reverse()	(abjad.tools.timetokentools.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker	reverse()	(abjad.tools.timetokentools.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker
method), 1801	method), 1966	method), 1966	method), 1966
rhythmtreetools.parse_rtm_syntax()	(in module abjad.tools.rhythmtreetools.parse_rtm_syntax),	rhythmtreetools.parse_rtm_syntax()	(in module abjad.tools.rhythmtreetools.parse_rtm_syntax),
1346	1346	1346	1346
rhythmtreetools.RhythmTreeContainer	(class in abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer)	rhythmtreetools.RhythmTreeContainer	(class in abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer)
1322	1322	1322	1322
rhythmtreetools.RhythmTreeLeaf	(class in abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf)	rhythmtreetools.RhythmTreeLeaf	(class in abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf)
1337	1337	1337	1337
rhythmtreetools.RhythmTreeNode	(class in abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode)	rhythmtreetools.RhythmTreeNode	(class in abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode)
1317	1317	1317	1317
rhythmtreetools.RhythmTreeParser	(class in abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser)	rhythmtreetools.RhythmTreeParser	(class in abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser)
1343	1343	1343	1343
roman_numeral_string	(abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree	roman_numeral_string	(abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree
attribute), 1819	attribute), 1819	attribute), 1819	attribute), 1819
root	(abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree	root	(abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree
attribute), 1425	attribute), 1425	attribute), 1425	attribute), 1425
root	(abjad.tools.sequencetools.Tree.Tree.Tree	root	(abjad.tools.sequencetools.Tree.Tree.Tree
attribute), 1438	attribute), 1438	attribute), 1438	attribute), 1438
root	(abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass	root	(abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass
attribute), 1804	attribute), 1804	attribute), 1804	attribute), 1804

attribute), 1050
rows (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix.CyclicMatrix attribute), 1421
rows (abjad.tools.sequencetools.Matrix.Matrix.Matrix attribute), 1434
rtm_format (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer attribute), 1328
rtm_format (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1341
rtm_format (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode attribute), 1321
S
scale_by_rational() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1709
scale_by_rational() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
scale_by_rational() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin method), 1705
scale_by_rational() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1718
scale_by_rational() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1738
scale_degree (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction attribute), 1822
scale_degree_to_named_chromatic_pitch_class() (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1817
scale_to_rational() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1709
scale_to_rational() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1702
scale_to_rational() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin method), 1705
scale_to_rational() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1719
scale_to_rational() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1739
scan_bare_word() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition method), 2041
scan_escaped_word() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition method), 2041
schemetools.format_scheme_value() (in module abjad.tools.schemetools.format_scheme_value), 1358
schemetools.Scheme (class in abjad.tools.schemetools.Scheme.Scheme), 1347
schemetools.SchemeAssociativeList (class in abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList), 1349
schemetools.SchemeColor (class in abjad.tools.schemetools.SchemeColor.SchemeColor), 1350
schemetools.SchemeMoment (class in abjad.tools.schemetools.SchemeMoment.SchemeMoment), 1352
schemetools.SchemePair (class in abjad.tools.schemetools.SchemePair.SchemePair), 1353
schemetools.SchemeVector (class in abjad.tools.schemetools.SchemeVector.SchemeVector), 1355
schemetools.SchemeVectorConstant (class in abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant), 1357
scoretemplatetools.GroupedRhythmicStavesScoreTemplate (class in abjad.tools.scoretemplatetools.GroupedRhythmicStavesScoreTemplate), 1360
scoretemplatetools.GroupedStavesScoreTemplate (class in abjad.tools.scoretemplatetools.GroupedStavesScoreTemplate), 1362
scoretemplatetools.ScoreTemplate (class in abjad.tools.scoretemplatetools.ScoreTemplate), 1359
scoretemplatetools.StringQuartetScoreTemplate (class in abjad.tools.scoretemplatetools.StringQuartetScoreTemplate), 1364
scoretemplatetools.TwoStaffPianoScoreTemplate (class in abjad.tools.scoretemplatetools.TwoStaffPianoScoreTemplate), 1366
scoretools.add_double_bar_to_end_of_score() (in module abjad.tools.scoretools.add_double_bar_to_end_of_score), 1411
scoretools.add_markup_to_end_of_score() (in module abjad.tools.scoretools.add_markup_to_end_of_score), 1412
scoretools.all_are_scores() (in module abjad.tools.scoretools.all_are_scores), 1412
scoretools.get_first_score_in_improper_parentage_of_component()

(in module abjad.tools.scoretools.get_first_score_in_improper_parentage_of_components), 1413

scoretools.get_first_score_in_proper_parentage_of_components (in module abjad.tools.scoretools.get_first_score_in_proper_parentage_of_components), 1413

scoretools.GrandStaff (class in abjad.tools.scoretools.GrandStaff), 1368

scoretools.InstrumentationSpecifier (class in abjad.tools.scoretools.InstrumentationSpecifier), 1377

scoretools.list_performer_names() (in module abjad.tools.scoretools.list_performer_names), 1413

scoretools.list_primary_performer_names() (in module abjad.tools.scoretools.list_primary_performer_names), 1414

scoretools.make_empty_piano_score() (in module abjad.tools.scoretools.make_empty_piano_score), 1415

scoretools.make_piano_score_from_leaves() (in module abjad.tools.scoretools.make_piano_score_from_leaves), 1415

scoretools.make_piano_sketch_score_from_leaves() (in module abjad.tools.scoretools.make_piano_sketch_score_from_leaves), 1416

scoretools.Performer (class in abjad.tools.scoretools.Performer), 1379

scoretools.PerformerInventory (class in abjad.tools.scoretools.PerformerInventory), 1382

scoretools.PianoStaff (class in abjad.tools.scoretools.PianoStaff), 1385

scoretools.Score (class in abjad.tools.scoretools.Score), 1394

scoretools.StaffGroup (class in abjad.tools.scoretools.StaffGroup), 1403

scripting_group (abjad.tools.abjadbooktools.AbjadBookScriptingGroup), 1887

scripting_group (abjad.tools.developerscripttools.AbjDevScriptingGroup), 1916

scripting_group (abjad.tools.developerscripttools.AbjGrepScriptingGroup), 1918

scripting_group (abjad.tools.developerscripttools.BuildApiScriptingGroup), 1921

scripting_group (abjad.tools.developerscripttools.CleanScriptingGroup), 1923

scripting_group (abjad.tools.developerscripttools.CountLinewidthsScriptingGroup), 1925

scripting_group (abjad.tools.developerscripttools.CountToolsScriptingGroup), 1928

scripting_group (abjad.tools.developerscripttools.DeveloperScriptingGroup), 1911

scripting_group (abjad.tools.developerscripttools.DirectoryScriptingGroup), 1913

scripting_group (abjad.tools.developerscripttools.MakeNewClassTemplateScriptingGroup), 1930

scripting_group (abjad.tools.developerscripttools.MakeNewFunctionTemplateScriptingGroup), 1932

scripting_group (abjad.tools.developerscripttools.RenameModulesScriptingGroup), 1935

scripting_group (abjad.tools.developerscripttools.ReplaceInFilesScriptingGroup), 1937

scripting_group (abjad.tools.developerscripttools.ReplacePromptsScriptingGroup), 1940

scripting_group (abjad.tools.developerscripttools.RunDoctestsScriptingGroup), 1942

scripting_group (abjad.tools.developerscripttools.SvnAddAllScriptingGroup), 1945

scripting_group (abjad.tools.developerscripttools.SvnCommitScriptingGroup), 1947

scripting_group (abjad.tools.developerscripttools.SvnMessageScriptingGroup), 1949

scripting_group (abjad.tools.developerscripttools.SvnUpdateScriptingGroup), 1952

semitones (abjad.tools.pitchtools.Accidental.Accidental), 1130

semitones (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject), 1066

semitones (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject), 1072

semitones (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject), 1077

semitones (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval), 1131

semitones (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval), 1143

semitones (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval), 1146

semitones (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject), 1084

semitones (abjad.tools.pitchtools.IntervalObject.IntervalObject), 1092

semitones (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval), 1175

semitones (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval), 1189

semitones (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval), 1194

semitones (abjad.tools.pitchtools.MelodicIntervalObject.MelodicIntervalObject), 1194

attribute), 1102

send_signal() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1971

sequencetools.all_are_assignable_integers() (in module abjad.tools.sequencetools.all_are_assignable_integers), 1444

sequencetools.all_are_equal() (in module abjad.tools.sequencetools.all_are_equal), 1444

sequencetools.all_are_integer_equivalent_exprs() (in module abjad.tools.sequencetools.all_are_integer_equivalent_exprs), 1445

sequencetools.all_are_integer_equivalent_numbers() (in module abjad.tools.sequencetools.all_are_integer_equivalent_numbers), 1445

sequencetools.all_are_nonnegative_integer_equivalent_numbers() (in module abjad.tools.sequencetools.all_are_nonnegative_integer_equivalent_numbers), 1445

sequencetools.all_are_nonnegative_integer_powers_of_two() (in module abjad.tools.sequencetools.all_are_nonnegative_integer_powers_of_two), 1446

sequencetools.all_are_nonnegative_integers() (in module abjad.tools.sequencetools.all_are_nonnegative_integers), 1446

sequencetools.all_are_numbers() (in module abjad.tools.sequencetools.all_are_numbers), 1446

sequencetools.all_are_pairs() (in module abjad.tools.sequencetools.all_are_pairs), 1447

sequencetools.all_are_pairs_of_types() (in module abjad.tools.sequencetools.all_are_pairs_of_types), 1447

sequencetools.all_are_positive_integer_equivalent_numbers() (in module abjad.tools.sequencetools.all_are_positive_integer_equivalent_numbers), 1447

sequencetools.all_are_positive_integers() (in module abjad.tools.sequencetools.all_are_positive_integers), 1448

sequencetools.all_are_unequal() (in module abjad.tools.sequencetools.all_are_unequal), 1448

sequencetools.count_length_two_runs_in_sequence() (in module abjad.tools.sequencetools.count_length_two_runs_in_sequence), 1448

sequencetools.CyclicList (class in abjad.tools.sequencetools.CyclicList.CyclicList), 1417

sequencetools.CyclicMatrix (class in abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix), 1420

sequencetools.CyclicTree (class in abjad.tools.sequencetools.CyclicTree.CyclicTree), 1423

sequencetools.CyclicTuple (class in abjad.tools.sequencetools.CyclicTuple.CyclicTuple), 1431

sequencetools.divide_sequence_elements_by_greatest_common_divisor() (in module abjad.tools.sequencetools.divide_sequence_elements_by_greatest_common_divisor), 1449

sequencetools.flatten_sequence() (in module abjad.tools.sequencetools.flatten_sequence), 1449

sequencetools.flatten_sequence_at_indices() (in module abjad.tools.sequencetools.flatten_sequence_at_indices), 1449

sequencetools.get_indices_of_sequence_elements_equal_to_true() (in module abjad.tools.sequencetools.get_indices_of_sequence_elements_equal_to_true), 1450

sequencetools.get_sequence_degree_of_rotational_symmetry() (in module abjad.tools.sequencetools.get_sequence_degree_of_rotational_symmetry), 1450

sequencetools.get_sequence_element_at_cyclic_index() (in module abjad.tools.sequencetools.get_sequence_element_at_cyclic_index), 1450

sequencetools.get_sequence_elements_at_indices() (in module abjad.tools.sequencetools.get_sequence_elements_at_indices), 1451

sequencetools.get_sequence_elements_frequency_distribution() (in module abjad.tools.sequencetools.get_sequence_elements_frequency_distribution), 1451

sequencetools.get_sequence_period_of_rotation() (in module abjad.tools.sequencetools.get_sequence_period_of_rotation), 1452

sequencetools.increase_sequence_elements_at_indices_by_addenda() (in module abjad.tools.sequencetools.increase_sequence_elements_at_indices_by_addenda), 1452

sequencetools.increase_sequence_elements_cyclically_by_addenda() (in module abjad.tools.sequencetools.increase_sequence_elements_cyclically_by_addenda), 1452

sequencetools.interlace_sequences() (in module abjad.tools.sequencetools.interlace_sequences), 1452

[1453](#)
 sequencetools.is_fraction_equivalent_pair()
 (in module ab-
 jad.tools.sequencetools.is_fraction_equivalent_pair(),
[1453](#)
 sequencetools.is_integer_equivalent_n_tuple()
 (in module ab-
 jad.tools.sequencetools.is_integer_equivalent_n_tuple(),
[1453](#)
 sequencetools.is_integer_equivalent_pair()
 (in module ab-
 jad.tools.sequencetools.is_integer_equivalent_pair(),
[1453](#)
 sequencetools.is_integer_equivalent_singleton()
 (in module ab-
 jad.tools.sequencetools.is_integer_equivalent_singleton(),
[1454](#)
 sequencetools.is_integer_n_tuple() (in module ab-
 jad.tools.sequencetools.is_integer_n_tuple(),
[1454](#)
 sequencetools.is_integer_pair() (in module ab-
 jad.tools.sequencetools.is_integer_pair(),
[1454](#)
 sequencetools.is_integer_singleton() (in module ab-
 jad.tools.sequencetools.is_integer_singleton(),
[1455](#)
 sequencetools.is_monotonically_decreasing_sequence()
 (in module ab-
 jad.tools.sequencetools.is_monotonically_decreasing_sequence(),
[1455](#)
 sequencetools.is_monotonically_increasing_sequence()
 (in module ab-
 jad.tools.sequencetools.is_monotonically_increasing_sequence(),
[1456](#)
 sequencetools.is_n_tuple() (in module ab-
 jad.tools.sequencetools.is_n_tuple), [1456](#)
 sequencetools.is_null_tuple() (in module ab-
 jad.tools.sequencetools.is_null_tuple), [1457](#)
 sequencetools.is_pair() (in module ab-
 jad.tools.sequencetools.is_pair), [1457](#)
 sequencetools.is_permutation() (in module ab-
 jad.tools.sequencetools.is_permutation),
[1457](#)
 sequencetools.is_repetition_free_sequence()
 (in module ab-
 jad.tools.sequencetools.is_repetition_free_sequence),
[1458](#)
 sequencetools.is_restricted_growth_function()
 (in module ab-
 jad.tools.sequencetools.is_restricted_growth_function),
[1458](#)
 sequencetools.is_singleton() (in module ab-
 jad.tools.sequencetools.is_singleton), [1459](#)
 sequencetools.is_strictly_decreasing_sequence()

(in module ab-
 jad.tools.sequencetools.is_strictly_decreasing_sequence),
[1459](#)
 sequencetools.is_strictly_increasing_sequence()
 (in module ab-
 jad.tools.sequencetools.is_strictly_increasing_sequence),
[1460](#)
 sequencetools.iterate_sequence_cyclically()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_cyclically),
[1460](#)
 sequencetools.iterate_sequence_cyclically_from_start_to_stop()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_cyclically_from_start_to_stop),
[1461](#)
 sequencetools.iterate_sequence_forward_and_backward_nonoverlapping()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_forward_and_backward_nonoverlapping),
[1461](#)
 sequencetools.iterate_sequence_forward_and_backward_overlapping()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_forward_and_backward_overlapping),
[1462](#)
 sequencetools.iterate_sequence_nwise_cyclic()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_nwise_cyclic),
[1462](#)
 sequencetools.iterate_sequence_nwise_strict()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_nwise_strict),
[1462](#)
 sequencetools.iterate_sequence_nwise_wrapped()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_nwise_wrapped),
[1463](#)
 sequencetools.iterate_sequence_pairwise_cyclic()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_pairwise_cyclic),
[1463](#)
 sequencetools.iterate_sequence_pairwise_strict()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_pairwise_strict),
[1463](#)
 sequencetools.iterate_sequence_pairwise_wrapped()
 (in module ab-
 jad.tools.sequencetools.iterate_sequence_pairwise_wrapped),
[1464](#)
 sequencetools.join_subsequences() (in module ab-
 jad.tools.sequencetools.join_subsequences),
[1464](#)
 sequencetools.join_subsequences_by_sign_of_subsequence_elements()
 (in module ab-
 jad.tools.sequencetools.join_subsequences_by_sign_of_subsequence_elements),
[1464](#)

sequencetools.map_sequence_elements_to_canonic_tuples((in module ab- jad.tools.sequencetools.map_sequence_elements_to_canonic_tuples), 1464	sequencetools.partition_sequence_by_restricted_growth_function((in module ab- jad.tools.sequencetools.partition_sequence_by_restricted_growth_function), 1470
sequencetools.map_sequence_elements_to_numbered_sublists((in module ab- jad.tools.sequencetools.map_sequence_elements_to_numbered_sublists), 1465	sequencetools.partition_sequence_by_sign_of_elements((in module ab- jad.tools.sequencetools.partition_sequence_by_sign_of_elements), 1470
sequencetools.Matrix (class in ab- jad.tools.sequencetools.Matrix.Matrix), 1433	sequencetools.partition_sequence_by_value_of_elements((in module ab- jad.tools.sequencetools.partition_sequence_by_value_of_elements), 1471
sequencetools.merge_duration_sequences((in module ab- jad.tools.sequencetools.merge_duration_sequences), 1465	sequencetools.partition_sequence_by_weights_at_least((in module ab- jad.tools.sequencetools.partition_sequence_by_weights_at_least), 1471
sequencetools.negate_absolute_value_of_sequence_elements_at_indices((in module ab- jad.tools.sequencetools.negate_absolute_value_of_sequence_elements_at_indices), 1465	sequencetools.partition_sequence_by_weights_at_most((in module ab- jad.tools.sequencetools.partition_sequence_by_weights_at_most), 1472
sequencetools.negate_absolute_value_of_sequence_elements_cyclically((in module ab- jad.tools.sequencetools.negate_absolute_value_of_sequence_elements_cyclically), 1466	sequencetools.partition_sequence_cyclically((in module ab- jad.tools.sequencetools.partition_sequence_cyclically), 1472
sequencetools.negate_sequence_elements_at_indices((in module ab- jad.tools.sequencetools.negate_sequence_elements_at_indices), 1466	sequencetools.partition_sequence_extended_to_counts((in module ab- jad.tools.sequencetools.partition_sequence_extended_to_counts), 1473
sequencetools.negate_sequence_elements_cyclically((in module ab- jad.tools.sequencetools.negate_sequence_elements_cyclically), 1466	sequencetools.permute_sequence() (in module ab- jad.tools.sequencetools.permute_sequence), 1473
sequencetools.override_sequence_elements_at_indices((in module ab- jad.tools.sequencetools.override_sequence_elements_at_indices), 1467	sequencetools.remove_sequence_elements_at_indices((in module ab- jad.tools.sequencetools.remove_sequence_elements_at_indices), 1473
sequencetools.pair_duration_sequence_elements_with_input_pair_value((in module ab- jad.tools.sequencetools.pair_duration_sequence_elements_with_input_pair_value), 1467	sequencetools.remove_sequence_elements_at_indices_cyclically((in module ab- jad.tools.sequencetools.remove_sequence_elements_at_indices_cyclically), 1474
sequencetools.partition_sequence_by_backgrounded_weights((in module ab- jad.tools.sequencetools.partition_sequence_by_backgrounded_weights), 1467	sequencetools.remove_subsequence_of_weight_at_index((in module ab- jad.tools.sequencetools.remove_subsequence_of_weight_at_index), 1474
sequencetools.partition_sequence_by_counts((in module ab- jad.tools.sequencetools.partition_sequence_by_counts), 1468	sequencetools.repeat_runs_in_sequence_to_count((in module ab- jad.tools.sequencetools.repeat_runs_in_sequence_to_count), 1474
sequencetools.partition_sequence_by_ratio_of_lengths((in module ab- jad.tools.sequencetools.partition_sequence_by_ratio_of_lengths), 1469	sequencetools.repeat_sequence_elements_at_indices((in module ab- jad.tools.sequencetools.repeat_sequence_elements_at_indices), 1475
sequencetools.partition_sequence_by_ratio_of_weights((in module ab- jad.tools.sequencetools.partition_sequence_by_ratio_of_weights), 1469	sequencetools.repeat_sequence_elements_at_indices_cyclically((in module ab- jad.tools.sequencetools.repeat_sequence_elements_at_indices_cyclically), 1475

[1475](#) `sequencetools.split_sequence_extended_to_weights()`
`sequencetools.repeat_sequence_elements_n_times_each()` (in module `ab-`
`jad.tools.sequencetools.split_sequence_extended_to_weights()`,
[1476](#) `sequencetools.sum_consecutive_sequence_elements_by_sign()`
`sequencetools.repeat_sequence_n_times()` (in module `ab-`
`jad.tools.sequencetools.repeat_sequence_n_times()`, `jad.tools.sequencetools.sum_consecutive_sequence_elements_by_`
[1476](#) [1481](#) `sequencetools.sum_sequence_elements_at_indices()`
`sequencetools.repeat_sequence_to_length()` (in module `ab-`
`jad.tools.sequencetools.repeat_sequence_to_length()`, `jad.tools.sequencetools.sum_sequence_elements_at_indices()`,
[1476](#) [1482](#) `sequencetools.Tree` (class in `ab-`
`sequencetools.repeat_sequence_to_weight_at_least()` (in module `ab-`
`jad.tools.sequencetools.Tree.Tree`), [1436](#)
[1477](#) `sequencetools.truncate_runs_in_sequence()`
`sequencetools.repeat_sequence_to_weight_at_most()` (in module `ab-`
`jad.tools.sequencetools.truncate_runs_in_sequence()`,
[1477](#) [1482](#) `sequencetools.truncate_sequence_to_sum()`
`sequencetools.repeat_sequence_to_weight_exactly()` (in module `ab-`
`jad.tools.sequencetools.truncate_sequence_to_sum()`,
[1477](#) [1482](#) `sequencetools.truncate_sequence_to_weight()`
`sequencetools.replace_sequence_elements_cyclically_with_new_material()` (in module `ab-`
`jad.tools.sequencetools.replace_sequence_elements_cyclically_with_new_material()`,
[1477](#) [1483](#) `sequencetools.yield_all_combinations_of_sequence_elements()`
`sequencetools.retain_sequence_elements_at_indices()` (in module `ab-`
`jad.tools.sequencetools.yield_all_combinations_of_sequence_ele`
[1477](#) [1484](#) `sequencetools.yield_all_k_ary_sequences_of_length()`
`sequencetools.retain_sequence_elements_at_indices_cyclically()` (in module `ab-`
`jad.tools.sequencetools.yield_all_k_ary_sequences_of_length()`,
[1478](#) [1484](#) `sequencetools.yield_all_pairs_between_sequences()`
`sequencetools.reverse_sequence()` (in module `ab-`
`jad.tools.sequencetools.reverse_sequence()`,
[1478](#) [1485](#) `sequencetools.yield_all_partitions_of_sequence()`
`sequencetools.reverse_sequence_elements()` (in module `ab-`
`jad.tools.sequencetools.yield_all_partitions_of_sequence()`,
[1479](#) [1485](#) `sequencetools.yield_all_permutations_of_sequence()`
`sequencetools.rotate_sequence()` (in module `ab-`
`jad.tools.sequencetools.rotate_sequence()`, `jad.tools.sequencetools.yield_all_permutations_of_sequence()`,
[1479](#) [1485](#) `sequencetools.yield_all_permutations_of_sequence_in_orbit()`
`sequencetools.splice_new_elements_between_sequence_elements()` (in module `ab-`
`jad.tools.sequencetools.splice_new_elements_between_sequence_elements()`, `jad.tools.sequencetools.yield_all_permutations_of_sequence_in_c`
[1479](#) [1486](#) `sequencetools.yield_all_restricted_growth_functions_of_length()`
`sequencetools.split_sequence_by_weights()` (in module `ab-`
`jad.tools.sequencetools.split_sequence_by_weights()`, `jad.tools.sequencetools.yield_all_restricted_growth_functions_of_`
[1480](#) [1486](#)

attribute), 1612
 set (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner attribute), 1619
 set (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner attribute), 1625
 set (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner attribute), 1632
 set (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1656
 set (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1665
 set (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute), 1685
 set (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1831
 set (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1841
 set (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1867
 setdefault() (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig method), 1896
 setdefault() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method), 1903
 setdefault() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1967
 setdefault() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1135
 setdefault() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector method), 1165
 setdefault() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector method), 1172
 setdefault() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 1181
 setdefault() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector method), 1222
 setdefault() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector method), 1245
 setdefault() (abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector method), 1115
 setdefault() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1709
 setdefault() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1741
 setup_argument_parser() (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript.AbjadBookScript method), 1887
 setup_argument_parser() (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript method), 1916
 setup_argument_parser() (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript method), 1918
 setup_argument_parser() (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript method), 1921
 setup_argument_parser() (abjad.tools.developerscripttools.CountLinewidthsScript.CountLinewidthsScript.CountLinewidthsScript method), 1926
 setup_argument_parser() (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScript method), 1928
 setup_argument_parser() (abjad.tools.developerscripttools.DeveloperScript.DeveloperScript.DeveloperScript method), 1911
 setup_argument_parser() (abjad.tools.developerscripttools.DirectoryScript.DirectoryScript.DirectoryScript method), 1913
 setup_argument_parser() (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript method), 1930
 setup_argument_parser() (abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript method), 1932
 setup_argument_parser() (abjad.tools.developerscripttools.RenameModulesScript.RenameModulesScript.RenameModulesScript method), 1935
 setup_argument_parser() (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript method), 1937
 setup_argument_parser() (abjad.tools.developerscripttools.ReplacePromptsScript.ReplacePromptsScript.ReplacePromptsScript method), 1940
 setup_argument_parser() (abjad.tools.developerscripttools.RunDoctestsScript.RunDoctestsScript.RunDoctestsScript method), 1947
 setup_argument_parser() (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript method), 1949
 setup_argument_parser() (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript method), 1952
 shape_string (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1537
 shape_string (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1558
 shift_by_rational() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1709

short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass	jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo		
attribute), 579	attribute), 689		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet	jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone		
attribute), 585	attribute), 695		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute	jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone		
attribute), 591	attribute), 701		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon	jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice		
attribute), 603	attribute), 707		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone	jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone		
attribute), 597	attribute), 713		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice	jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone		
attribute), 609	attribute), 719		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet	jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice		
attribute), 615	attribute), 725		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn	jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet		
attribute), 621	attribute), 731		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.Flute.Flute.Flute	jad.tools.instrumenttools.Tuba.Tuba.Tuba		
attribute), 627	attribute), 737		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn	jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion		
attribute), 632	attribute), 743		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel	jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone		
attribute), 638	attribute), 749		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.Guitar.Guitar.Guitar	jad.tools.instrumenttools.Viola.Viola.Viola		
attribute), 644	attribute), 755		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.Harp.Harp.Harp	jad.tools.instrumenttools.Violin.Violin.Violin		
attribute), 650	attribute), 761		
short_instrument_name	(ab-	short_instrument_name	(ab-
jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord	jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone		
attribute), 656	attribute), 767		
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Marimba.Marimba.Marimba	jad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark		
attribute), 665	attribute), 400		
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice	jad.tools.instrumenttools.Accordion.Accordion.Accordion		
attribute), 671	attribute), 489		
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Oboe.Oboe.Oboe	jad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute		
attribute), 677	attribute), 495		
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Piano.Piano.Piano	jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone		
attribute), 683	attribute), 501		

short_instrument_name_markup jad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 507	(ab- short_instrument_name_markup attribute), 615	(ab- short_instrument_name_markup attribute), 615
short_instrument_name_markup jad.tools.instrumenttools.BaronetSaxophone.BaronetSaxophone.BaronetSaxophone attribute), 519	(ab- short_instrument_name_markup attribute), 621	(ab- short_instrument_name_markup attribute), 621
short_instrument_name_markup jad.tools.instrumenttools.BaronetVoice.BaronetVoice.BaronetVoice attribute), 525	(ab- short_instrument_name_markup attribute), 627	(ab- short_instrument_name_markup attribute), 627
short_instrument_name_markup jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 531	(ab- short_instrument_name_markup attribute), 632	(ab- short_instrument_name_markup attribute), 632
short_instrument_name_markup jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 537	(ab- short_instrument_name_markup attribute), 638	(ab- short_instrument_name_markup attribute), 638
short_instrument_name_markup jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 561	(ab- short_instrument_name_markup attribute), 644	(ab- short_instrument_name_markup attribute), 644
short_instrument_name_markup jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 543	(ab- short_instrument_name_markup attribute), 650	(ab- short_instrument_name_markup attribute), 650
short_instrument_name_markup jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 549	(ab- short_instrument_name_markup attribute), 656	(ab- short_instrument_name_markup attribute), 656
short_instrument_name_markup jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 555	(ab- short_instrument_name_markup attribute), 665	(ab- short_instrument_name_markup attribute), 665
short_instrument_name_markup jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 513	(ab- short_instrument_name_markup attribute), 671	(ab- short_instrument_name_markup attribute), 671
short_instrument_name_markup jad.tools.instrumenttools.Cello.Cello.Cello attribute), 567	(ab- short_instrument_name_markup attribute), 677	(ab- short_instrument_name_markup attribute), 677
short_instrument_name_markup jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute), 573	(ab- short_instrument_name_markup attribute), 683	(ab- short_instrument_name_markup attribute), 683
short_instrument_name_markup jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass attribute), 579	(ab- short_instrument_name_markup attribute), 689	(ab- short_instrument_name_markup attribute), 689
short_instrument_name_markup jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet attribute), 585	(ab- short_instrument_name_markup attribute), 695	(ab- short_instrument_name_markup attribute), 695
short_instrument_name_markup jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute attribute), 591	(ab- short_instrument_name_markup attribute), 701	(ab- short_instrument_name_markup attribute), 701
short_instrument_name_markup jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon attribute), 603	(ab- short_instrument_name_markup attribute), 707	(ab- short_instrument_name_markup attribute), 707
short_instrument_name_markup jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone attribute), 597	(ab- short_instrument_name_markup attribute), 713	(ab- short_instrument_name_markup attribute), 713
short_instrument_name_markup jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice attribute), 609	(ab- short_instrument_name_markup attribute), 719	(ab- short_instrument_name_markup attribute), 719

short_instrument_name_markup	(abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 1937	jad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript attribute), 1937
short_instrument_name_markup	(abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 731	skipped_files (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript attribute), 1937
short_instrument_name_markup	(abjad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 737	skiptools.all_are_skips() (in module abjad.tools.skiptools.all_are_skips), 1498
short_instrument_name_markup	(abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion attribute), 743	skiptools.make_repeated_skips_from_time_signature() (in module abjad.tools.skiptools.make_repeated_skips_from_time_signature), 1498
short_instrument_name_markup	(abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 749	skiptools.make_repeated_skips_from_time_signatures() (in module abjad.tools.skiptools.make_repeated_skips_from_time_signatures), 1498
short_instrument_name_markup	(abjad.tools.instrumenttools.Viola.Viola.Viola attribute), 755	skiptools.make_skips_with_multiplied_durations() (in module abjad.tools.skiptools.make_skips_with_multiplied_durations), 1499
short_instrument_name_markup	(abjad.tools.instrumenttools.Violin.Violin.Violin attribute), 761	skiptools.replace_leaves_in_expr_with_skips() (in module abjad.tools.skiptools.replace_leaves_in_expr_with_skips), 1499
short_instrument_name_markup	(abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 767	Skip (class in abjad.tools.skiptools.Skip.Skip), 1495
sievetools.all_are_residue_class_expressions()	(in module abjad.tools.sievetools.all_are_residue_class_expressions), 1494	skiptools.yield_groups_of_skips_in_sequence() (in module abjad.tools.skiptools.yield_groups_of_skips_in_sequence), 1499
sievetools.cycle_tokens_to_sieve()	(in module abjad.tools.sievetools.cycle_tokens_to_sieve), 1494	slashedGraceContainer() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy method), 2035
sievetools.ResidueClass	(class in abjad.tools.sievetools.ResidueClass.ResidueClass), 1490	slope (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment attribute), 1183
sievetools.ResidueClassExpression	(class in abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression), 1492	sort() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 381
signature (abjad.tools.timeintervaltools.TimeInterval.TimeInterval attribute), 1707	sort() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval attribute), 1701	sort() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 415
signature (abjad.tools.timeintervaltools.TimeIntervalAggregator.TimeIntervalAggregator attribute), 1701	sort() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin attribute), 1704	sort() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1906
signature (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree attribute), 1712	sort() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree attribute), 1726	sort() (abjad.tools.timeintervaltools.TimeIntervalAggregator.TimeIntervalAggregator attribute), 1701
signature (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1726	sort() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1726	sort() (abjad.tools.timeintervaltools.TimeIntervalAggregator.TimeIntervalAggregator attribute), 1701
size (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1050	sort() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1726	sort() (abjad.tools.timeintervaltools.TimeIntervalAggregator.TimeIntervalAggregator attribute), 1701
skip() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy method), 2035	sort() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1726	sort() (abjad.tools.timeintervaltools.TimeIntervalAggregator.TimeIntervalAggregator attribute), 1701
skip_rendering (abjad.tools.abjadbooktools.AbjadBookProcessor.AbjadBookProcessor attribute), 1885	sort() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1726	sort() (abjad.tools.timeintervaltools.TimeIntervalAggregator.TimeIntervalAggregator attribute), 1701
skipped_directories	(abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1726	sort() (abjad.tools.timeintervaltools.TimeIntervalAggregator.TimeIntervalAggregator attribute), 1701

method), 1252
 sort() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.
 method), 1257
 sort() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory.
 method), 1263
 sort() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory.
 method), 1383
 sort() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList.
 method), 1418
 sounding_pitch (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic.
 attribute), 1033
 sounding_pitch (abjad.tools.notetools.Note.Note.Note attribute,
 attribute), 1038
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Accordion.Accordion.Accordion
 attribute), 489
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute
 attribute), 496
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone.
 attribute), 502
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone.
 attribute), 508
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.BaronSaxophone.BaronSaxophone.BaronSaxophone.
 attribute), 520
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.BaronVoice.BaronVoice.BaronVoice.
 attribute), 525
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet.
 attribute), 532
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute
 attribute), 538
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon
 attribute), 562
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone.
 attribute), 544
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone.
 attribute), 550
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice
 attribute), 555
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet.
 attribute), 514
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Cello.Cello.Cello
 attribute), 567
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA.
 attribute), 579
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass.
 attribute), 579
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet.
 attribute), 592
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon.
 attribute), 604
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.
 attribute), 598
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice.
 attribute), 609
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet.
 attribute), 616
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn.
 attribute), 622
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Flute.Flute.Flute
 attribute), 627
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn.
 attribute), 633
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel.
 attribute), 638
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Guitar.Guitar.Guitar
 attribute), 644
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Harp.Harp.Harp
 attribute), 651
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord.
 attribute), 657
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Marimba.Marimba.Marimba
 attribute), 665
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.
 attribute), 671
 sounding_pitch_of_fingered_middle_c (ab-
 jad.tools.instrumenttools.Oboe.Oboe.Oboe
 attribute), 671

attribute), 678

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Piano.Piano.Piano attribute), 684

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 690

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone attribute), 696

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 702

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 707

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 714

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 720

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 725

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 731

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 737

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion attribute), 743

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 749

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Viola.Viola.Viola attribute), 755

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Violin.Violin.Violin attribute), 761

sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 767

sounding_pitches (abjad.tools.chordtools.Chord.Chord.Chord attribute), 267

source_pitch_range (abjad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent attribute), 1254

span (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 241

span (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 250

spanners (abjad.tools.chordtools.Chord.Chord.Chord attribute), 268

spanners (abjad.tools.componenttools.Component.Component.Component attribute), 276

spanners (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 336

spanners (abjad.tools.containertools.Container.Container.Container attribute), 347

spanners (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer attribute), 350

spanners (abjad.tools.containertools.SopranoSaxophoneContext.SopranoSaxophoneContext.SopranoSaxophoneContext attribute), 386

spanners (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer attribute), 476

spanners (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 829

spanners (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure attribute), 981

spanners (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 996

spanners (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1004

spanners (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 1034

spanners (abjad.tools.notetools.Note.Note.Note attribute), 1038

spanners (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1309

spanners (abjad.tools.resttools.Rest.Rest.Rest attribute), 1312

spanners (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1377

spanners (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1387

spanners (abjad.tools.scoretools.Score.Score.Score attribute), 1396

spanners (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1405

spanners (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1496

spanners (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1656

spanners (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1665

spanners (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1831

spanners (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1841

spanners (abjad.tools.transpositiontools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent attribute), 1867

spanned_complex_beam_spanners (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 241

spanned_complex_beam_spanners (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 250

spannertools.apply_octavation_spanner_to_pitched_components()	(in module ab-
jad.tools.spannertools.apply_octavation_spanner_to_pitched_components),	jad.tools.spannertools.get_spanners_attached_to_any_improper_p
1638	spannertools.get_spanners_attached_to_any_proper_child_of_component()
spannertools.BracketSpanner (class in ab-	(in module ab-
jad.tools.spannertools.BracketSpanner.BracketSpanner),	jad.tools.spannertools.get_spanners_attached_to_any_proper_chi
1512	1643
spannertools.ComplexGlissandoSpanner (class in ab-	spannertools.get_spanners_attached_to_any_proper_parent_of_component()
jad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner),	module ab-
1519	jad.tools.spannertools.get_spanners_attached_to_any_proper_parc
spannertools.CrescendoSpanner (class in ab-	1644
jad.tools.spannertools.CrescendoSpanner.CrescendoSpanner),	spannertools.get_spanners_attached_to_component()
1526	(in module ab-
spannertools.DecrescendoSpanner (class in ab-	jad.tools.spannertools.get_spanners_attached_to_component),
jad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner),	1534
1534	spannertools.get_spanners_contained_by_components()
spannertools.destroy_spanners_attached_to_component()	(in module ab-
(in module ab-	jad.tools.spannertools.get_spanners_contained_by_components),
jad.tools.spannertools.destroy_spanners_attached_to_component),	1638
1638	spannertools.get_spanners_covered_by_components()
spannertools.destroy_spanners_attached_to_components_in_expr()	(in module ab-
(in module ab-	jad.tools.spannertools.get_spanners_covered_by_components),
jad.tools.spannertools.destroy_spanners_attached_to_components_in_expr),	1639
1639	spannertools.get_spanners_on_components_or_component_children()
spannertoolsDirectedSpanner (class in ab-	(in module ab-
jad.tools.spannertoolsDirectedSpannerDirectedSpanner),	jad.tools.spannertools.get_spanners_on_components_or_compon
1500	1645
spannertools.DynamicTextSpanner (class in ab-	spannertools.get_spanners_that_cross_components()
jad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner),	module ab-
1543	jad.tools.spannertools.get_spanners_that_cross_components),
spannertools.find_index_of_spanner_component_at_score_offset()	1646
(in module ab-	spannertools.get_spanners_that_dominat
jad.tools.spannertools.find_index_of_spanner_component_at_score_offset),	module ab-
1639	jad.tools.spannertools.get_spanners_that_dominat
spannertools.find_spanner_component_starting_at_exactly_score_offset()	1646
(in module ab-	spannertools.get_spanners_that_dominat
jad.tools.spannertools.find_spanner_component_starting_at_exactly_score_offset),	module ab-
1640	jad.tools.spannertools.get_spanners_that_dominat
spannertools.fracture_spanners_attached_to_component()	1646
(in module ab-	spannertools.get_spanners_that_dominat
jad.tools.spannertools.fracture_spanners_attached_to_component),	module ab-
1640	jad.tools.spannertools.get_spanners_that_dominat
spannertools.fracture_spanners_that_cross_components()	1646
(in module ab-	spannertools.get_the_only_spanner_attached_to_any_improper_parent_of_
jad.tools.spannertools.fracture_spanners_that_cross_components),	module ab-
1641	jad.tools.spannertools.get_the_only_spanner_attached_to_any_im
spannertools.get_nth_leaf_in_spanner()	(in module ab-
jad.tools.spannertools.get_nth_leaf_in_spanner),	spannertools.get_the_only_spanner_attached_to_component()
1641	(in module ab-
spannertools.get_spanners_attached_to_any_improper_child_of_component()	jad.tools.spannertools.get_the_only_spanner_attached_to_compo
(in module ab-	1647
jad.tools.spannertools.get_spanners_attached_to_any_improper_child_of_component),	spannertools.GlissandoSpanner (class in ab-
1642	jad.tools.spannertools.GlissandoSpanner.GlissandoSpanner),
spannertools.get_spanners_attached_to_any_improper_parent_of_component()	1540

staff_spaces (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject.DiatonicIntervalObject.TimeSignatureMark.TimeSignatureMark attribute), 1077

staff_spaces (abjad.tools.pitchtools.HarmonicDiatonicIntervals.HarmonicDiatonicIntervals.HarmonicDiatonicIntervals.Accordion.Accordion attribute), 1146

staff_spaces (abjad.tools.pitchtools.MelodicDiatonicIntervals.MelodicDiatonicIntervals.MelodicDiatonicIntervals.Flute.Flute.Flute attribute), 1194

stafftools.all_are_staves() (in module abjad.tools.stafftools.all_are_staves), 1671

stafftools.get_first_staff_in_improper_parentage_of_component() (in module abjad.tools.stafftools.get_first_staff_in_improper_parentage_of_component), 1672

stafftools.get_first_staff_in_proper_parentage_of_component() (in module abjad.tools.stafftools.get_first_staff_in_proper_parentage_of_component), 1672

stafftools.make_rhythmic_sketch_staff() (in module abjad.tools.stafftools.make_rhythmic_sketch_staff), 1672

stafftools.RhythmicStaff (class in abjad.tools.stafftools.RhythmicStaff.RhythmicStaff), 1654

stafftools.Staff (class in abjad.tools.stafftools.Staff.Staff), 1663

start (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 1586

start (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 1707

start (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.instrumenttools.Cello.Cello.Cello attribute), 1701

start (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.TimeIntervalMixin.instrumenttools.Clarinets.Clarinets.Clarinets attribute), 1705

start (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree.instrumenttools.Contrabass.Contrabass.Contrabass attribute), 1712

start (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.instrumenttools.Clarinets.Clarinets.Clarinets attribute), 1726

start (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute attribute), 1820

start_cells (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon attribute), 1055

start_component (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone attribute), 378

start_component (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice attribute), 393

start_component (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute), 396

start_component (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 399

start_component (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark.instrumenttools.Flute.Flute.Flute attribute), 403

start_component (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 407

start_component (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 411

start_component (abjad.tools.instrumenttools.Guitar.Guitar.Guitar.start_component (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment), 643
 attribute), 643
 start_component (abjad.tools.instrumenttools.Harp.Harp.Harp.start_component (abjad.tools.marktools.Mark.Mark.Mark.Mark), 649
 attribute), 649
 start_component (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord.start_component (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo), 655
 attribute), 655
 start_component (abjad.tools.instrumenttools.Marimba.Marimba.Marimba.start_component (abjad.tools.markuptools.Markup.Markup.Markup.Markup), 664
 attribute), 664
 start_component (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice.start_component (abjad.tools.marktools.VerticalMoment.VerticalMoment.VerticalMoment), 670
 attribute), 670
 start_component (abjad.tools.instrumenttools.Oboe.Oboe.Oboe.start_dynamic_string (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner), 676
 attribute), 676
 start_component (abjad.tools.instrumenttools.Piano.Piano.Piano.start_dynamic_string (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner), 682
 attribute), 682
 start_component (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo.start_dynamic_string (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner), 688
 attribute), 688
 start_component (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone.start_dynamic_string (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner), 694
 attribute), 694
 start_component (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone.start_leaves (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment), 700
 attribute), 700
 start_component (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.start_notes (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment), 706
 attribute), 706
 start_component (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.start_offset (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner), 712
 attribute), 712
 start_component (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.start_offset (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner), 718
 attribute), 718
 start_component (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice.start_offset (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner), 724
 attribute), 724
 start_component (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet.start_offset (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner), 730
 attribute), 730
 start_component (abjad.tools.instrumenttools.Tuba.Tuba.Tuba.start_offset (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner), 736
 attribute), 736
 start_component (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.start_offset (abjad.tools.chordtools.Chord.Chord.Chord), 742
 attribute), 742
 start_component (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone.start_offset (abjad.tools.componenttools.Component.Component.Component), 748
 attribute), 748
 start_component (abjad.tools.instrumenttools.Viola.Viola.Viola.start_offset (abjad.tools.containertools.Cluster.Cluster.Cluster), 754
 attribute), 754
 start_component (abjad.tools.instrumenttools.Violin.Violin.Violin.start_offset (abjad.tools.containertools.Container.Container.Container), 760
 attribute), 760
 start_component (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone.start_offset (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer), 766
 attribute), 766
 start_component (abjad.tools.marktools.Annotation.Annotation.Annotation.start_offset (abjad.tools.contexttools.Context.Context.Context), 891
 attribute), 891
 start_component (abjad.tools.marktools.Articulation.Articulation.Articulation.start_offset (abjad.tools.durationtools.TimespanConstant.TimespanConstant.TimespanConstant), 894
 attribute), 894
 start_component (abjad.tools.marktools.BarLine.BarLine.BarLine.start_offset (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer), 897
 attribute), 897
 start_component (abjad.tools.marktools.BendAfter.BendAfter.BendAfter.start_offset (abjad.tools.leaftools.Leaf.Leaf.Leaf), 900
 attribute), 900
 start_component (abjad.tools.marktools.DirectedMark.DirectedMark.DirectedMark.start_offset (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure), 888
 attribute), 888
 start_component (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark.start_offset (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure), 903
 attribute), 903

[illegible]

attribute), 1034

start_offset_in_seconds (abjad.tools.notetools.Note.Note.Note attribute), 1039

start_offset_in_seconds (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest attribute), 1309

start_offset_in_seconds (abjad.tools.resttools.Rest.Rest.Rest attribute), 1312

start_offset_in_seconds (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1370

start_offset_in_seconds (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1387

start_offset_in_seconds (abjad.tools.scoretools.Score.Score.Score attribute), 1396

start_offset_in_seconds (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1405

start_offset_in_seconds (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1496

start_offset_in_seconds (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1656

start_offset_in_seconds (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1665

start_offset_in_seconds (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1832

start_offset_in_seconds (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1842

start_offset_in_seconds (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1867

start_pitch (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), 1261

start_pitch_is_included_in_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), 1261

start_pitches (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn attribute), 1055

starting_line_number (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock attribute), 1889

stop (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner attribute), 1586

stop (abjad.tools.timeintervaltools.TimeInterval.TimeInterval attribute), 1708

stop (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin attribute), 1701

stop (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin attribute), 1705

stop (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree attribute), 1726

stop (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1726

stop (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator attribute), 1820

stop_cells (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn attribute), 1055

stop_dynamic_string (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner attribute), 1529

stop_dynamic_string (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner attribute), 1537

stop_dynamic_string (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner attribute), 1559

stop_offset (abjad.tools.beamtools.BeamSpanner.BeamSpanner attribute), 225

stop_offset (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner attribute), 232

stop_offset (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 240

stop_offset (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 249

stop_offset (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner attribute), 257

stop_offset (abjad.tools.chordtools.Chord.Chord.Chord attribute), 276

stop_offset (abjad.tools.componenttools.Component.Component attribute), 276

stop_offset (abjad.tools.containertools.Cluster.Cluster attribute), 337

stop_offset (abjad.tools.containertools.Container.Container attribute), 343

stop_offset (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer attribute), 351

stop_offset (abjad.tools.contexttools.Context.Context attribute), 386

stop_offset (abjad.tools.durationtools.TimespanConstant.TimespanConstant attribute), 453

stop_offset (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn attribute), 476

stop_offset (abjad.tools.leaftools.Leaf.Leaf attribute), 829

stop_offset (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 993

stop_offset (abjad.tools.measuretools.Measure.Measure attribute), 993

attribute), 1004

stop_offset (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 1034

stop_offset (abjad.tools.notetools.Note.Note.Note attribute), 1039

stop_offset (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1309

stop_offset (abjad.tools.resttools.Rest.Rest.Rest attribute), 1312

stop_offset (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1371

stop_offset (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1387

stop_offset (abjad.tools.scoretools.Score.Score.Score attribute), 1396

stop_offset (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1405

stop_offset (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1496

stop_offset (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner attribute), 1514

stop_offset (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner attribute), 1520

stop_offset (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1528

stop_offset (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1536

stop_offset (abjad.tools.spannertoolsDirectedSpannerDirectedSpannerDirectedSpanner attribute), 1501

stop_offset (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner attribute), 1544

stop_offset (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner attribute), 1551

stop_offset (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute), 1558

stop_offset (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner attribute), 1565

stop_offset (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner attribute), 1572

stop_offset (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner attribute), 1578

stop_offset (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner attribute), 1585

stop_offset (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner attribute), 1592

stop_offset (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner attribute), 1599

stop_offset (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner attribute), 1605

stop_offset (abjad.tools.spannertools.Spanner.Spanner.Spanner attribute), 1507

stop_offset (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner attribute), 1612

stop_offset (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner attribute), 1619

stop_offset (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner attribute), 1626

stop_offset (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner attribute), 1632

stop_offset (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner attribute), 1656

stop_offset (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner attribute), 1665

stop_offset (abjad.tools.spannertools.TieSpanner.TieSpanner.TieSpanner attribute), 1685

stop_offset (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1832

stop_offset (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1842

stop_offset (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1867

stop_offset_in_seconds (abjad.tools.chordtools.Chord.Chord.Chord attribute), 268

stop_offset_in_seconds (abjad.tools.chordtools.Chord.Chord.Chord attribute), 276

stop_offset_in_seconds (abjad.tools.chordtools.Chord.Chord.Chord attribute), 276

stop_offset_in_seconds (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 343

stop_offset_in_seconds (abjad.tools.containertools.FixedDurationContainer.FixedDurationContainer.FixedDurationContainer attribute), 386

stop_offset_in_seconds (abjad.tools.gracetools.GraceContainer.GraceContainer.GraceContainer attribute), 829

stop_offset_in_seconds (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure attribute), 993

stop_offset_in_seconds (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1004

stop_offset_in_seconds (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1034

stop_offset_in_seconds (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1034

jad.tools.notetools.Note.Note.Note attribute), 1881
1039 storage_format (abjad.tools.abctools.SortableAttributeEqualityAbjadObject
stop_offset_in_seconds (ab- attribute), 1882
jad.tools.resttools.MultiMeasureRest.MultiMeasureRest attribute), 1885
stop_offset_in_seconds (ab- storage_format (abjad.tools.abjadbooktools.AbjadBookProcessor.AbjadBook
attribute), 1309 attribute), 1885
stop_offset_in_seconds (ab- storage_format (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookSc
jad.tools.resttools.Rest.Rest.Rest attribute), 1887
1312 storage_format (abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBl
stop_offset_in_seconds (ab- attribute), 1889
jad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff storage_format (abjad.tools.abjadbooktools.HTMLOutputFormat.HTMLOu
attribute), 1371 attribute), 1890
stop_offset_in_seconds (ab- storage_format (abjad.tools.abjadbooktools.LaTeXOutputFormat.LaTeXOu
attribute), 1387 attribute), 1892
stop_offset_in_seconds (ab- storage_format (abjad.tools.abjadbooktools.OutputFormat.OutputFormat.O
attribute), 1396 attribute), 1884
stop_offset_in_seconds (ab- storage_format (abjad.tools.abjadbooktools.ReSTOutputFormat.ReSTOutpu
attribute), 1396 attribute), 1893
stop_offset_in_seconds (ab- storage_format (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSp
jad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 225
attribute), 1405 storage_format (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBea
stop_offset_in_seconds (ab- attribute), 232
jad.tools.skiptools.Skip.Skip.Skip attribute), storage_format (abjad.tools.beamtools.DuratedComplexBeamSpanner.Dura
1496 attribute), 240
stop_offset_in_seconds (ab- storage_format (abjad.tools.beamtools.MeasuredComplexBeamSpanner.Me
jad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 249
attribute), 1656 storage_format (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBea
stop_offset_in_seconds (ab- attribute), 257
jad.tools.stafftools.Staff.Staff.Staff attribute), storage_format (abjad.tools.chordtools.Chord.Chord.Chord
1665 attribute), 268
stop_offset_in_seconds (ab- storage_format (abjad.tools.componenttools.Component.Component.Comp
jad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 278
attribute), 1832 storage_format (abjad.tools.componenttools.ContainmentSignature.Contain
stop_offset_in_seconds (ab- attribute), 278
jad.tools.tuplettools.Tuplet.Tuplet.Tuplet storage_format (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.A
attribute), 1842 attribute), 1896
stop_offset_in_seconds (ab- storage_format (abjad.tools.containertools.Cluster.Cluster.Cluster
jad.tools.voicetools.Voice.Voice.Voice attribute), 337
1867 storage_format (abjad.tools.containertools.Container.Container.Container
stop_pitch (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), 343
attribute), 1261 storage_format (abjad.tools.containertools.FixedDurationContainer.FixedDu
stop_pitch_is_included_in_range (ab- attribute), 351
jad.tools.pitchtools.PitchRange.PitchRange.PitchRange storage_format (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark
attribute), 1261 attribute), 378
stop_pitches (abjad.tools.pitcharraytools.PitchArrayColumnsPitchArrayColumns attribute), 381
1055 storage_format (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInven
storage_format (abjad.tools.abctools.AbjadObject.AbjadObject storage_format (abjad.tools.contexttools.Context.Context.Context
attribute), 1875 attribute), 386
storage_format (abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject storage_format (abjad.tools.contexttools.ContextMarking.AbjadObjectContext
attribute), 1876 attribute), 393
storage_format (abjad.tools.abctools.ImmutableAbjadObjectsImmutableAbjadObjects storage_format (abjad.tools.contexttools.DynamicMark.DynamicMark.Dyn
attribute), 1877 attribute), 396
storage_format (abjad.tools.abctools.Parser.Parser.Parser storage_format (abjad.tools.contexttools.InstrumentMark.InstrumentMark.I
attribute), 1879 attribute), 399
storage_format (abjad.tools.abctools.ScoreSelection.ScoreSelection storage_format (abjad.tools.contexttools.KeySignatureMark.KeySignatureM

attribute), 403
storage_format (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark.tools.APICrawler.APICrawler.APICrawler attribute), 407
storage_format (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark.abjad.tools.documentationtools.ClassCrawler.ClassCrawler attribute), 412
storage_format (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter attribute), 415
storage_format (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark.tools.Documenter.Documenter.Documenter attribute), 420
storage_format (abjad.tools.datastructuretools.Digraph.Digraph.Digraph.abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler attribute), 1901
storage_format (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary.abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter attribute), 1903
storage_format (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph attribute), 1905
storage_format (abjad.tools.datastructuretools.OrdinalConstant.OrdinalConstant.OrdinalConstant.abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler attribute), 1909
storage_format (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript.AbjDevScript.abjad.tools.documentationtools.Pipe.Pipe.Pipe attribute), 1916
storage_format (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript.AbjGrepScript.abjad.tools.durationtools.Duration.Duration.Duration attribute), 1918
storage_format (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript.BuildApiScript.abjad.tools.offsettools.Offset.Offset.Offset attribute), 1921
storage_format (abjad.tools.developerscripttools.CleanScripts.CleanScripts.CleanScripts.abjad.tools.timespanconstanttools.TimespanConstant.TimespanConstant attribute), 1923
storage_format (abjad.tools.developerscripttools.CountLineWidthsScript.CountLineWidthsScript.CountLineWidthsScript.abjad.tools.containertools.Container.Container attribute), 1925
storage_format (abjad.tools.developerscripttools.CountToolsScript.CountToolsScript.CountToolsScript.abjad.tools.accordions.Accordions.Accordions attribute), 1928
storage_format (abjad.tools.developerscripttools.DeveloperScript.DeveloperScript.DeveloperScript.abjad.tools.altoflute.AltoFlute.AltoFlute attribute), 1911
storage_format (abjad.tools.developerscripttools.DirectoryScript.DirectoryScript.DirectoryScript.abjad.tools.altoflute.AltoFlute attribute), 1913
storage_format (abjad.tools.developerscripttools.MakeNewClassTemplateScript.MakeNewClassTemplateScript.MakeNewClassTemplateScript.abjad.tools.altoflute.AltoFlute attribute), 1930
storage_format (abjad.tools.developerscripttools.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.MakeNewFunctionTemplateScript.abjad.tools.altoflute.AltoFlute attribute), 1932
storage_format (abjad.tools.developerscripttools.RenameModuleScript.RenameModuleScript.RenameModuleScript.abjad.tools.altoflute.AltoFlute attribute), 1935
storage_format (abjad.tools.developerscripttools.ReplaceInFilesScript.ReplaceInFilesScript.ReplaceInFilesScript.abjad.tools.altoflute.AltoFlute attribute), 1937
storage_format (abjad.tools.developerscripttools.ReplacePrologueScript.ReplacePrologueScript.ReplacePrologueScript.abjad.tools.altoflute.AltoFlute attribute), 1940
storage_format (abjad.tools.developerscripttools.RunDoctestScript.RunDoctestScript.RunDoctestScript.abjad.tools.altoflute.AltoFlute attribute), 1942
storage_format (abjad.tools.developerscripttools.SvnAddAllScriptsScript.SvnAddAllScriptsScript.SvnAddAllScriptsScript.abjad.tools.altoflute.AltoFlute attribute), 1945
storage_format (abjad.tools.developerscripttools.SvnCommitScript.SvnCommitScript.SvnCommitScript.abjad.tools.altoflute.AltoFlute attribute), 1947
storage_format (abjad.tools.developerscripttools.SvnMessageScript.SvnMessageScript.SvnMessageScript.abjad.tools.altoflute.AltoFlute attribute), 1949
storage_format (abjad.tools.developerscripttools.SvnUpdateScript.SvnUpdateScript.SvnUpdateScript.abjad.tools.altoflute.AltoFlute attribute), 1952
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1955
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1954
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1957
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1959
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1960
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1962
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1964
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1966
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1968
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 1970
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 446
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 450
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 453
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 476
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 488
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 494
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 500
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 506
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 518
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 524
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 530
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 536
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 560
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 542
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 548
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 554
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 512
storage_format (abjad.tools.documentationtools.AbjAPIGenerator.AbjAPIGenerator.AbjAPIGenerator.abjad.tools.altoflute.AltoFlute attribute), 512

attribute), 566

storage_format (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA), 572

storage_format (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass), 578

storage_format (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet), 584

storage_format (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute), 590

storage_format (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon), 602

storage_format (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone), 596

storage_format (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice), 608

storage_format (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet), 614

storage_format (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn), 620

storage_format (abjad.tools.instrumenttools.Flute.Flute.Flute), 626

storage_format (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn), 631

storage_format (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel), 637

storage_format (abjad.tools.instrumenttools.Guitar.Guitar.Guitar), 643

storage_format (abjad.tools.instrumenttools.Harp.Harp.Harp), 649

storage_format (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord), 655

storage_format (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory), 658

storage_format (abjad.tools.instrumenttools.Marimba.Marimba.Marimba), 664

storage_format (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice), 670

storage_format (abjad.tools.instrumenttools.Oboe.Oboe.Oboe), 676

storage_format (abjad.tools.instrumenttools.Piano.Piano.Piano), 682

storage_format (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo), 688

storage_format (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone), 694

storage_format (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone), 700

storage_format (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice), 706

storage_format (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone), 712

storage_format (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone), 718

storage_format (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice), 724

storage_format (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet), 730

storage_format (abjad.tools.instrumenttools.Tuba.Tuba.Tuba), 736

storage_format (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion), 742

storage_format (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone), 748

storage_format (abjad.tools.instrumenttools.Viola.Viola.Viola), 754

storage_format (abjad.tools.instrumenttools.Violon.Violon.Violon), 760

storage_format (abjad.tools.instrumenttools.Violin.Violin.Violin), 766

storage_format (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication), 824

storage_format (abjad.tools.leaftools.Leaf.Leaf.Leaf), 829

storage_format (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken), 862

storage_format (abjad.tools.lilypondfiletools.DateTimeToken.DateTimeToken.DateTimeToken), 870

storage_format (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken), 878

storage_format (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken), 880

storage_format (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy), 2035

storage_format (abjad.tools.lilypondparsertools.LilyPondDuration.LilyPondDuration.LilyPondDuration), 2037

storage_format (abjad.tools.lilypondparsertools.LilyPondEvent.LilyPondEvent.LilyPondEvent), 2038

storage_format (abjad.tools.lilypondparsertools.LilyPondFraction.LilyPondFraction.LilyPondFraction), 2039

storage_format (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition.LilyPondLexicalDefinition), 2041

storage_format (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser), 2047

storage_format (abjad.tools.lilypondparsertools.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition.LilyPondSyntacticalDefinition), 2048

storage_format (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser.ReducedLyParser), 2070

storage_format (abjad.tools.lymanipulationtools.SchemeParser.SchemeParser.SchemeParser), 2075

storage_format (abjad.tools.lymanipulationtools.SyntaxNode.SyntaxNode.SyntaxNode), 2078

storage_format (abjad.tools.marktools.Annotation.Annotation.Annotation), 891

storage_format (abjad.tools.marktools.Articulation.Articulation.Articulation), 894

storage_format (abjad.tools.marktools.BarLine.BarLine.BarLine), 897

storage_format (abjad.tools.marktools.BendAfter.BendAfter.BendAfter), 897

attribute), 900
 storage_format (abjad.tools.marktools.DirectedMark.DirectedMark attribute), 888
 storage_format (abjad.tools.marktools.LilyPondCommandMarkup.LilyPondCommandMarkup attribute), 904
 storage_format (abjad.tools.marktools.LilyPondComment.LilyPondComment attribute), 907
 storage_format (abjad.tools.marktools.Mark.Mark.Mark attribute), 909
 storage_format (abjad.tools.marktools.StemTremolo.StemTremolo attribute), 911
 storage_format (abjad.tools.markuptools.Markup.Markup.Markup attribute), 933
 storage_format (abjad.tools.markuptools.MarkupCommand.MarkupCommand attribute), 935
 storage_format (abjad.tools.markuptools.MarkupInventory.MarkupInventory attribute), 937
 storage_format (abjad.tools.mathtools.BoundedObject.BoundedObject attribute), 945
 storage_format (abjad.tools.mathtools.NonreducedFraction.NonreducedFraction attribute), 948
 storage_format (abjad.tools.mathtools.Ratio.Ratio.Ratio attribute), 954
 storage_format (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 981
 storage_format (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 993
 storage_format (abjad.tools.measuretools.Measure.Measure attribute), 1004
 storage_format (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic attribute), 1034
 storage_format (abjad.tools.notetools.Note.Note.Note attribute), 1039
 storage_format (abjad.tools.notetools.NoteHead.NoteHead.NoteHead attribute), 1042
 storage_format (abjad.tools.pitcharraytools.PitchArray.PitchArray attribute), 1050
 storage_format (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell attribute), 1053
 storage_format (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn attribute), 1055
 storage_format (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow attribute), 1057
 storage_format (abjad.tools.pitchtools.Accidental.Accidental attribute), 1130
 storage_format (abjad.tools.pitchtools.ChromaticIntervalClassObject.ChromaticIntervalClassObject attribute), 1064
 storage_format (abjad.tools.pitchtools.ChromaticIntervalObject.ChromaticIntervalObject attribute), 1066
 storage_format (abjad.tools.pitchtools.ChromaticObject.ChromaticObject attribute), 1067
 storage_format (abjad.tools.pitchtools.ChromaticPitchObject.ChromaticPitchObject attribute), 1069
 storage_format (abjad.tools.pitchtools.CounterpointIntervalClassObject.CounterpointIntervalClassObject attribute), 1071
 storage_format (abjad.tools.pitchtools.CounterpointIntervalObject.CounterpointIntervalObject attribute), 1072
 storage_format (abjad.tools.pitchtools.CounterpointMarkupObject.CounterpointMarkupObject attribute), 1074
 storage_format (abjad.tools.pitchtools.DiatonicIntervalClassObject.DiatonicIntervalClassObject attribute), 1075
 storage_format (abjad.tools.pitchtools.DiatonicIntervalObject.DiatonicIntervalObject attribute), 1077
 storage_format (abjad.tools.pitchtools.DiatonicObject.DiatonicObject attribute), 1078
 storage_format (abjad.tools.pitchtools.DiatonicPitchClassObject.DiatonicPitchClassObject attribute), 1079
 storage_format (abjad.tools.pitchtools.DiatonicPitchObject.DiatonicPitchObject attribute), 1081
 storage_format (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval attribute), 1131
 storage_format (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass attribute), 1133
 storage_format (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector attribute), 1134
 storage_format (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment attribute), 1137
 storage_format (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet attribute), 1140
 storage_format (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval attribute), 1143
 storage_format (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass attribute), 1144
 storage_format (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval attribute), 1146
 storage_format (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass attribute), 1147
 storage_format (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet attribute), 1149
 storage_format (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment attribute), 1152
 storage_format (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet attribute), 1155
 storage_format (abjad.tools.pitchtools.HarmonicIntervalClassObject.HarmonicIntervalClassObject attribute), 1082
 storage_format (abjad.tools.pitchtools.HarmonicIntervalObject.HarmonicIntervalObject attribute), 1084
 storage_format (abjad.tools.pitchtools.HarmonicObject.HarmonicObject attribute), 1086
 storage_format (abjad.tools.pitchtools.IntervalClassObject.IntervalClassObject attribute), 1087
 storage_format (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet attribute), 1089
 storage_format (abjad.tools.pitchtools.IntervalObject.IntervalObject attribute), 1092
 storage_format (abjad.tools.pitchtools.IntervalObjectClass.IntervalObjectClass attribute), 1093
 storage_format (abjad.tools.pitchtools.IntervalObjectSegment.IntervalObjectSegment attribute), 1096

attribute), 1124
 storage_format (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet attribute), 1126
 storage_format (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), 1261
 storage_format (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory attribute), 1262
 storage_format (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow attribute), 1266
 storage_format (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1309
 storage_format (abjad.tools.resttools.Rest.Rest.Rest attribute), 1312
 storage_format (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer attribute), 1328
 storage_format (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1341
 storage_format (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode attribute), 1321
 storage_format (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser attribute), 1344
 storage_format (abjad.tools.schemetools.Scheme.Scheme.Scheme attribute), 1348
 storage_format (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList attribute), 1349
 storage_format (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor attribute), 1351
 storage_format (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment attribute), 1353
 storage_format (abjad.tools.schemetools.SchemePair.SchemePair.SchemePair attribute), 1354
 storage_format (abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector attribute), 1355
 storage_format (abjad.tools.schemetools.SchemeVectorConsistent.SchemeVectorConsistent.SchemeVectorConsistent attribute), 1357
 storage_format (abjad.tools.scoretemplatetools.GroupedRhythmicStaff.GroupedRhythmicStaff.GroupedRhythmicStaff attribute), 1361
 storage_format (abjad.tools.scoretemplatetools.GroupedStaves.GroupedStaves.GroupedStaves attribute), 1363
 storage_format (abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplate.ScoreTemplate attribute), 1359
 storage_format (abjad.tools.scoretemplatetools.StringQuarterNoteStaff.StringQuarterNoteStaff.StringQuarterNoteStaff attribute), 1365
 storage_format (abjad.tools.scoretemplatetools.TwoStaffPiano.TwoStaffPiano.TwoStaffPiano attribute), 1367
 storage_format (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1371
 storage_format (abjad.tools.scoretools.InstrumentationSpecification.InstrumentationSpecification.InstrumentationSpecification attribute), 1378
 storage_format (abjad.tools.scoretools.Performer.Performer.Performer attribute), 1380
 storage_format (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory attribute), 1382
 storage_format (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1388
 storage_format (abjad.tools.scoretools.PitchObjectSet.PitchObjectSet.PitchObjectSet attribute), 1396
 storage_format (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1405
 storage_format (abjad.tools.scoretools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory attribute), 1421
 storage_format (abjad.tools.scoretools.TwelveToneRow.TwelveToneRow.TwelveToneRow attribute), 1426
 storage_format (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix.CyclicMatrix attribute), 1434
 storage_format (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1439
 storage_format (abjad.tools.rhythmtreetools.RhythmTreeContainer.RhythmTreeContainer.RhythmTreeContainer attribute), 1491
 storage_format (abjad.tools.rhythmtreetools.RhythmTreeLeaf.RhythmTreeLeaf.RhythmTreeLeaf attribute), 1492
 storage_format (abjad.tools.rhythmtreetools.RhythmTreeNode.RhythmTreeNode.RhythmTreeNode attribute), 1496
 storage_format (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser.RhythmTreeParser attribute), 1514
 storage_format (abjad.tools.spannertools.ComplexGlissandoSpanner.ComplexGlissandoSpanner.ComplexGlissandoSpanner attribute), 1520
 storage_format (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1528
 storage_format (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1536
 storage_format (abjad.tools.spannertools.DirectedSpanner.DirectedSpanner.DirectedSpanner attribute), 1501
 storage_format (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner attribute), 1544
 storage_format (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner attribute), 1551
 storage_format (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute), 1558
 storage_format (abjad.tools.spannertools.GroupedRhythmicStaff.GroupedRhythmicStaff.GroupedRhythmicStaff attribute), 1565
 storage_format (abjad.tools.spannertools.GroupedStaves.GroupedStaves.GroupedStaves attribute), 1572
 storage_format (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner attribute), 1578
 storage_format (abjad.tools.spannertools.StringQuarterNoteStaff.StringQuarterNoteStaff.StringQuarterNoteStaff attribute), 1586
 storage_format (abjad.tools.spannertools.TwoStaffPiano.TwoStaffPiano.TwoStaffPiano attribute), 1592
 storage_format (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner attribute), 1599
 storage_format (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner attribute), 1606
 storage_format (abjad.tools.spannertools.Spanner.Spanner.Spanner attribute), 1507
 storage_format (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner attribute), 1612
 storage_format (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner attribute), 1612

attribute), 1619
 storage_format (abjad.tools.spannertools.TextSpanner.TextSpanner attribute), 1626
 storage_format (abjad.tools.spannertools.TrillSpanner.TrillSpanner attribute), 1632
 storage_format (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff attribute), 1656
 storage_format (abjad.tools.stafftools.Staff.Staff attribute), 1665
 storage_format (abjad.tools.tietools.TieChain.TieChain.TieChain attribute), 1683
 storage_format (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute), 1685
 storage_format (abjad.tools.timeintervaltools.TimeInterval.TimeInterval attribute), 1708
 storage_format (abjad.tools.timeintervaltools.TimeIntervalAggregation.TimeIntervalAggregation attribute), 1701
 storage_format (abjad.tools.timeintervaltools.TimeIntervalMeasure.TimeIntervalMeasure attribute), 1705
 storage_format (abjad.tools.timeintervaltools.TimeIntervalTool.TimeIntervalTool attribute), 1713
 storage_format (abjad.tools.timeintervaltools.TimeIntervalTool.TimeIntervalTool attribute), 1726
 storage_format (abjad.tools.timetoken tools.BurnishedTimeToken.BurnishedTimeToken attribute), 1758
 storage_format (abjad.tools.timetoken tools.EqualDivisionTimeToken.EqualDivisionTimeToken attribute), 1769
 storage_format (abjad.tools.timetoken tools.IncisedTimeToken.IncisedTimeToken attribute), 1760
 storage_format (abjad.tools.timetoken tools.NoteFilledTimeToken.NoteFilledTimeToken attribute), 1772
 storage_format (abjad.tools.timetoken tools.OutputBurnishedSignalFilledTimeToken.OutputBurnishedSignalFilledTimeToken attribute), 1775
 storage_format (abjad.tools.timetoken tools.OutputIncisedNoteFilledTimeToken.OutputIncisedNoteFilledTimeToken attribute), 1778
 storage_format (abjad.tools.timetoken tools.OutputIncisedRestFilledTimeToken.OutputIncisedRestFilledTimeToken attribute), 1781
 storage_format (abjad.tools.timetoken tools.OutputIncisedTimeToken.OutputIncisedTimeToken attribute), 1762
 storage_format (abjad.tools.timetoken tools.RestFilledTimeToken.RestFilledTimeToken attribute), 1784
 storage_format (abjad.tools.timetoken tools.SignalFilledTimeToken.SignalFilledTimeToken attribute), 1787
 storage_format (abjad.tools.timetoken tools.SkipFilledTimeToken.SkipFilledTimeToken attribute), 1789
 storage_format (abjad.tools.timetoken tools.TimeTokenMaker.TimeTokenMaker attribute), 1764
 storage_format (abjad.tools.timetoken tools.TokenBurnishedSignalFilledTimeToken.TokenBurnishedSignalFilledTimeToken attribute), 1792
 storage_format (abjad.tools.timetoken tools.TokenIncisedNoteFilledTimeToken.TokenIncisedNoteFilledTimeToken attribute), 1796
 storage_format (abjad.tools.timetoken tools.TokenIncisedRestFilledTimeToken.TokenIncisedRestFilledTimeToken attribute), 1799
 storage_format (abjad.tools.timetoken tools.TokenIncisedTimeToken.TokenIncisedTimeToken attribute), 1766
 storage_format (abjad.tools.timetoken tools.TupletMonadTimeTokenMaker.TupletMonadTimeTokenMaker attribute), 1801
 storage_format (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass attribute), 1804
 storage_format (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator attribute), 1808
 storage_format (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator attribute), 1809
 storage_format (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator attribute), 1811
 storage_format (abjad.tools.tonalitytools.Mode.Mode.Mode attribute), 1812
 storage_format (abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator attribute), 1813
 storage_format (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator attribute), 1814
 storage_format (abjad.tools.tonalitytools.MisScale.Scale.Scale attribute), 1817
 storage_format (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree attribute), 1819
 storage_format (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator attribute), 1820
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 1822
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 1832
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 1842
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 1860
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 1867
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2082
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2080
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2084
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2085
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2087
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2088
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2090
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2091
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2093
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2094
 storage_format (abjad.tools.tonalitytools.TokenMaker.TokenMaker attribute), 2094

attribute), 2096

storage_format(abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.lowercase_file_name_attribute), 2097

storage_format(abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.lowercase_file_name_attribute), 2099

storage_format(abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.lowercase_file_name_attribute), 2100

storage_format(abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.lowercase_file_name_attribute), 2102

storage_format(abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.lowercase_file_name_attribute), 2103

stringtools.arg_to_bidirectional_direction_string() (in module abjad.tools.stringtools.arg_to_bidirectional_direction_string), 1672

stringtools.arg_to_bidirectional_lilypond_symbol() (in module abjad.tools.stringtools.arg_to_bidirectional_lilypond_symbol), 1673

stringtools.arg_to_tridirectional_direction_string() (in module abjad.tools.stringtools.arg_to_tridirectional_direction_string), 1673

stringtools.arg_to_tridirectional_lilypond_symbol() (in module abjad.tools.stringtools.arg_to_tridirectional_lilypond_symbol), 1674

stringtools.arg_to_tridirectional_ordinal_constant() (in module abjad.tools.stringtools.arg_to_tridirectional_ordinal_constant), 1674

stringtools.capitalize_string_start() (in module abjad.tools.stringtools.capitalize_string_start), 1675

stringtools.format_input_lines_as_doc_string() (in module abjad.tools.stringtools.format_input_lines_as_doc_string), 1675

stringtools.format_input_lines_as_regression_test() (in module abjad.tools.stringtools.format_input_lines_as_regression_test), 1675

stringtools.is_lowercamelcase_string() (in module abjad.tools.stringtools.is_lowercamelcase_string), 1676

stringtools.is_space_delimited_lowercase_string() (in module abjad.tools.stringtools.is_space_delimited_lowercase_string), 1677

stringtools.is_underscore_delimited_lowercase_file_name() (in module abjad.tools.stringtools.is_underscore_delimited_lowercase_file_name), 1677

stringtools.is_underscore_delimited_lowercase_file_name_with_extensions() (in module abjad.tools.stringtools.is_underscore_delimited_lowercase_file_name_with_extensions), 1677

stringtools.is_uppercamelcase_string() (in module abjad.tools.stringtools.is_uppercamelcase_string), 1678

stringtools.space_delimited_lowercase_to_uppercamelcase() (in module abjad.tools.stringtools.space_delimited_lowercase_to_uppercamelcase), 1678

stringtools.string_to_strict_directory_name() (in module abjad.tools.stringtools.string_to_strict_directory_name), 1679

stringtools.strip_diacritics_from_binary_string() (in module abjad.tools.stringtools.strip_diacritics_from_binary_string), 1679

stringtools.underscore_delimited_lowercase_to_lowercamelcase() (in module abjad.tools.stringtools.underscore_delimited_lowercase_to_lowercamelcase), 1679

stringtools.underscore_delimited_lowercase_to_uppercamelcase() (in module abjad.tools.stringtools.underscore_delimited_lowercase_to_uppercamelcase), 1679

stringtools.uppercamelcase_to_space_delimited_lowercase() (in module abjad.tools.stringtools.uppercamelcase_to_space_delimited_lowercase), 1680

stringtools.uppercamelcase_to_underscore_delimited_lowercase() (in module abjad.tools.stringtools.uppercamelcase_to_underscore_delimited_lowercase), 1680

strip_prompt(abjad.tools.abjadbooktools.CodeBlock.CodeBlock.CodeBlock.lowercase_file_name_attribute), 1889

style(abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner.lowercase_file_name_attribute), 1599

subdominant(abjad.tools.tonalitytools.Scale.Scale.Scale.lowercase_file_name_attribute), 1817

submediant(abjad.tools.tonalitytools.Scale.Scale.Scale.lowercase_file_name_attribute), 1817

suono_reale(abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic.lowercase_file_name_attribute), 1034

superdominant(abjad.tools.tonalitytools.Scale.Scale.Scale.lowercase_file_name_attribute), 1817

[suppress_meter \(abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute\), 983](#)
[suppress_meter \(abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute\), 995](#)
[suspension \(abjad.tools.tonalitytools.TonalFunction.TonalFunction attribute\), 1822](#)
[symbolic_accidental_string \(abjad.tools.pitchtools.Accidental.Accidental attribute\), 1130](#)
[symbolic_string \(abjad.tools.tonalitytools.ScaleDegree.ScaleDegree attribute\), 1819](#)
[symbolic_string \(abjad.tools.tonalitytools.TonalFunction.TonalFunction attribute\), 1822](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method\), 1140](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method\), 1150](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method\), 1155](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet method\), 1090](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet method\), 1098](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method\), 1162](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method\), 1187](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method\), 1200](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method\), 1213](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet method\), 1219](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method\), 1241](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.ObjectSet.ObjectSet method\), 1112](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet method\), 1121](#)
[symmetric_difference\(\) \(abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet method\), 1121](#)

method), 2041		jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition
t_INITIAL_markup_notes_222()	(ab-	method), 2042
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		method), 2042
t_INITIAL_markup_notes_227()	(ab-	t_markup_548() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition.LilyPondLexicalDefinition
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		t_markup_601() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition
method), 2041		method), 2042
t_INITIAL_markup_notes_353()	(ab-	method), 2042
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		method), 2042
t_INITIAL_markup_notes_353_boolean()	(ab-	t_newline() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition.LilyPondLexicalDefinition
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		t_newline() (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser
method), 2041		method), 2072
t_INITIAL_markup_notes_353_identifier()	(ab-	method), 2076
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		t_newline() (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser
t_INITIAL_notes_233()	(ab-	t_notes_417() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		method), 2042
t_INITIAL_notes_387()	(ab-	method), 2042
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		method), 2042
t_INITIAL_notes_390()	(ab-	t_notes_424() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		t_notes_428() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition
method), 2042		method), 2042
t_INITIAL_notes_396()	(ab-	method), 2042
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		method), 2042
t_INITIAL_notes_399()	(ab-	t_notes_433() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		t_notes_error() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition
method), 2042		method), 2042
t_INITIAL_notes_686()	(ab-	method), 2042
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		method), 2072
t_INTEGER() (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser		method), 2076
method), 2076		method), 2043
t_INTEGER() (abjad.tools.rhythmtreetools.RhythmTreeParser.RhythmTreeParser		method), 2043
method), 1345		method), 2076
t_INTEGER_N() (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser		method), 2043
method), 2072		method), 2077
t_INTEGER_P() (abjad.tools.lilypondparsertools.ReducedLyParser.ReducedLyParser		method), 2077
method), 2072		method), 2077
t_L_PAREN() (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser		method), 2077
method), 2076		t_quote_446() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition
t_longcomment_291()	(ab-	t_quote_446() (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		method), 2077
t_longcomment_293()	(ab-	method), 2043
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		method), 2043
t_longcomment_296()	(ab-	t_quote_456() (abjad.tools.lilypondparsertools.SchemeParser.SchemeParser
jad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition		t_quote_error() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition.LilyPondLexicalDefinition
method), 2042		method), 2043
t_longcomment_error()	(ab-	method), 2043

t_quote_error() (abjad.tools.lilypondparsertools.SchemeParser, 2077
 method), 2077
 t_quote_XXX() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition, 2043
 method), 2043
 t_R_PAREN() (abjad.tools.lilypondparsertools.SchemeParser, 2076
 method), 2076
 t_scheme_error() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition, 2043
 method), 2043
 t_version_242() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition, 2043
 method), 2043
 t_version_278() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition, 2043
 method), 2043
 t_version_341() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition, 2043
 method), 2043
 t_version_error() (abjad.tools.lilypondparsertools.LilyPondLexicalDefinition, 2043
 method), 2043
 t_whitespace() (abjad.tools.lilypondparsertools.SchemeParser, 2077
 method), 2077
 target_context (abjad.tools.contexttools.ClefMark, 378
 attribute), 378
 target_context (abjad.tools.contexttools.ContextMark, 393
 attribute), 393
 target_context (abjad.tools.contexttools.DynamicMark, 396
 attribute), 396
 target_context (abjad.tools.contexttools.InstrumentMark, 399
 attribute), 399
 target_context (abjad.tools.contexttools.KeySignatureMark, 403
 attribute), 403
 target_context (abjad.tools.contexttools.StaffChangeMark, 407
 attribute), 407
 target_context (abjad.tools.contexttools.TempoMark, 412
 attribute), 412
 target_context (abjad.tools.contexttools.TimeSignatureMark, 420
 attribute), 420
 target_context (abjad.tools.instrumenttools.Accordion, 488
 attribute), 488
 target_context (abjad.tools.instrumenttools.AltoFlute, 494
 attribute), 494
 target_context (abjad.tools.instrumenttools.AltoSaxophone, 500
 attribute), 500
 target_context (abjad.tools.instrumenttools.AltoTrombone, 506
 attribute), 506
 target_context (abjad.tools.instrumenttools.BaritoneSaxophone, 518
 attribute), 518
 target_context (abjad.tools.instrumenttools.BaritoneVoice, 524
 attribute), 524
 target_context (abjad.tools.instrumenttools.BassClarinet, 530
 attribute), 530
 target_context (abjad.tools.instrumenttools.BassFlute, 536
 attribute), 536
 target_context (abjad.tools.instrumenttools.Bassoon, 560
 attribute), 560
 target_context (abjad.tools.instrumenttools.BassSaxophone, 542
 attribute), 542
 target_context (abjad.tools.instrumenttools.BassTrombone, 548
 attribute), 548
 target_context (abjad.tools.instrumenttools.BassVoice, 554
 attribute), 554
 target_context (abjad.tools.instrumenttools.BFlatClarinet, 512
 attribute), 512
 target_context (abjad.tools.instrumenttools.Cello, 566
 attribute), 566
 target_context (abjad.tools.instrumenttools.ClayPott, 572
 attribute), 572
 target_context (abjad.tools.instrumenttools.ColyPott, 578
 attribute), 578
 target_context (abjad.tools.instrumenttools.ColyPott, 584
 attribute), 584
 target_context (abjad.tools.instrumenttools.ColyPott, 590
 attribute), 590
 target_context (abjad.tools.instrumenttools.Contrabassoon, 602
 attribute), 602
 target_context (abjad.tools.instrumenttools.ContrabassSaxophone, 596
 attribute), 596
 target_context (abjad.tools.instrumenttools.ContraltoVoice, 608
 attribute), 608
 target_context (abjad.tools.instrumenttools.EFlatClarinet, 614
 attribute), 614
 target_context (abjad.tools.instrumenttools.EnglishHorn, 620
 attribute), 620
 target_context (abjad.tools.instrumenttools.Flute, 626
 attribute), 626
 target_context (abjad.tools.instrumenttools.FrenchHorn, 631
 attribute), 631
 target_context (abjad.tools.instrumenttools.Glockenspiel, 637
 attribute), 637
 target_context (abjad.tools.instrumenttools.Guitar, 643
 attribute), 643
 target_context (abjad.tools.instrumenttools.Harp, 649
 attribute), 649
 target_context (abjad.tools.instrumenttools.Harpsichord, 655
 attribute), 655
 target_context (abjad.tools.instrumenttools.Marimba, 664
 attribute), 664
 target_context (abjad.tools.instrumenttools.MezzoSopranoVoice, 670
 attribute), 670
 target_context (abjad.tools.instrumenttools.Oboe, 676
 attribute), 676
 target_context (abjad.tools.instrumenttools.Piano, 682
 attribute), 682
 target_context (abjad.tools.instrumenttools.Piccolo, 688
 attribute), 688
 target_context (abjad.tools.instrumenttools.SopraninoSaxophone, 694
 attribute), 694
 target_context (abjad.tools.instrumenttools.SopranoSaxophone, 700
 attribute), 700
 target_context (abjad.tools.instrumenttools.SopranoVoice, 706
 attribute), 706

jad.tools.tietools.TieSpanner.TieSpanner),
1684
time() (abjad.tools.lilypondparsertools.GuileProxy.GuileProxy.GuileProxy.calculate_sustain_centroid_of_intervals()
method), 2035
timeintervaltools.all_are_intervals_or_trees_or_empty()
(in module ab- 1747
jad.tools.timeintervaltools.all_are_intervals_or_trees_or_empty()
1744
timeintervaltools.all_intervals_are_contiguous()
(in module ab- 1747
jad.tools.timeintervaltools.all_intervals_are_contiguous()
1745
timeintervaltools.all_intervals_are_nonoverlapping()
(in module ab- 1747
jad.tools.timeintervaltools.all_intervals_are_nonoverlapping()
1745
timeintervaltools.calculate_density_of_attacks_in_interval()
(in module ab- 1747
jad.tools.timeintervaltools.calculate_density_of_attacks_in_interval()
1745
timeintervaltools.calculate_density_of_releases_in_interval()
(in module ab- 1748
jad.tools.timeintervaltools.calculate_density_of_releases_in_interval()
1745
timeintervaltools.calculate_depth_centroid_of_intervals()
(in module ab- 1748
jad.tools.timeintervaltools.calculate_depth_centroid_of_intervals()
1745
timeintervaltools.calculate_depth_centroid_of_intervals_in_interval()
(in module ab- 1748
jad.tools.timeintervaltools.calculate_depth_centroid_of_intervals_in_interval()
1745
timeintervaltools.calculate_depth_density_of_intervals()
(in module ab- 1748
jad.tools.timeintervaltools.calculate_depth_density_of_intervals()
1745
timeintervaltools.calculate_depth_density_of_intervals_in_interval()
(in module ab- 1748
jad.tools.timeintervaltools.calculate_depth_density_of_intervals_in_interval()
1746
timeintervaltools.calculate_mean_attack_of_intervals()
(in module ab- 1748
jad.tools.timeintervaltools.calculate_mean_attack_of_intervals()
1746
timeintervaltools.calculate_mean_release_of_intervals()
(in module ab- 1749
jad.tools.timeintervaltools.calculate_mean_release_of_intervals()
1746
timeintervaltools.calculate_min_mean_and_max_depth_of_intervals()
(in module ab- 1749
jad.tools.timeintervaltools.calculate_min_mean_and_max_depth_of_intervals()
1746
timeintervaltools.calculate_min_mean_and_max_durations_of_intervals()
(in module ab- 1749
jad.tools.timeintervaltools.calculate_min_mean_and_max_durations_of_intervals()
1349
timeintervaltools.explode_intervals_compactly()

timeintervaltools.explode_intervals_compactly()	(in module ab- 1749	timeintervaltools.explode_intervals_compactly()	(in module ab- 1752
timeintervaltools.explode_intervals_into_n_trees_heuristically()	(in module ab- 1749	timeintervaltools.explode_intervals_into_n_trees_heuristically()	(in module ab- 1752
timeintervaltools.explode_intervals_uncompactly()	(in module ab- 1749	timeintervaltools.explode_intervals_uncompactly()	(in module ab- 1753
timeintervaltools.fuse_overlapping_intervals()	(in module ab- 1750	timeintervaltools.fuse_overlapping_intervals()	(in module ab- 1753
timeintervaltools.fuse_tangent_or_overlapping_intervals()	(in module ab- 1750	timeintervaltools.fuse_tangent_or_overlapping_intervals()	(in module ab- 1753
timeintervaltools.get_all_unique_bounds_in_intervals()	(in module ab- 1750	timeintervaltools.get_all_unique_bounds_in_intervals()	(in module ab- 1754
timeintervaltools.group_overlapping_intervals_and_yield_groups()	(in module ab- 1750	timeintervaltools.group_overlapping_intervals_and_yield_groups()	(in module ab- 1754
timeintervaltools.group_tangent_or_overlapping_intervals_and_yield_groups()	(in module ab- 1751	timeintervaltools.group_tangent_or_overlapping_intervals_and_yield_groups()	(in module ab- 1755
timeintervaltools.make_monophonic_percussion_score_from_nonoverlapping_intervals()	(in module ab- 1751	timeintervaltools.make_monophonic_percussion_score_from_nonoverlapping_intervals()	(in module ab- 1755
timeintervaltools.make_polyphonic_percussion_score_from_nonoverlapping_trees()	(in module ab- 1751	timeintervaltools.make_polyphonic_percussion_score_from_nonoverlapping_trees()	(in module ab- 1756
timeintervaltools.make_voice_from_nonoverlapping_intervals()	(in module ab- 1751	timeintervaltools.make_voice_from_nonoverlapping_intervals()	(in module ab- 1700
timeintervaltools.mask_intervals_with_intervals()	(in module ab- 1751	timeintervaltools.mask_intervals_with_intervals()	(in module ab- 1704
timeintervaltools.resolve_overlaps_between_nonoverlapping_trees()	(in module ab- 1752	timeintervaltools.resolve_overlaps_between_nonoverlapping_trees()	(in module ab- 1704
timeintervaltools.round_interval_bounds_to_nearest_multiple_of_rational()	(in module ab- 1752	timeintervaltools.round_interval_bounds_to_nearest_multiple_of_rational()	(in module ab- 1704

1823
 tonalitytools.analyze_tonal_function() (in module ab-
 jad.tools.tonalitytools.analyze_tonal_function),
 1824
 tonalitytools.are_scalar_notes() (in module ab-
 jad.tools.tonalitytools.are_scalar_notes),
 1824
 tonalitytools.are_stepwise_ascending_notes()
 (in module ab-
 jad.tools.tonalitytools.are_stepwise_ascending_notes),
 1824
 tonalitytools.are_stepwise_descending_notes()
 (in module ab-
 jad.tools.tonalitytools.are_stepwise_descending_notes),
 1825
 tonalitytools.are_stepwise_notes() (in module ab-
 jad.tools.tonalitytools.are_stepwise_notes),
 1825
 tonalitytools.chord_class_cardinality_to_extent()
 (in module ab-
 jad.tools.tonalitytools.chord_class_cardinality_to_extent),
 1825
 tonalitytools.chord_class_extent_to_cardinality()
 (in module ab-
 jad.tools.tonalitytools.chord_class_extent_to_cardinality),
 1826
 tonalitytools.chord_class_extent_to_extent_name()
 (in module ab-
 jad.tools.tonalitytools.chord_class_extent_to_extent_name),
 1826
 tonalitytools.ChordClass (class in ab-
 jad.tools.tonalitytools.ChordClass.ChordClass),
 1803
 tonalitytools.ChordQualityIndicator (class in ab-
 jad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator),
 1807
 tonalitytools.diatonic_interval_class_segment_to_chord_quality_string()
 (in module ab-
 jad.tools.tonalitytools.diatonic_interval_class_segment_to_chord_quality_string),
 1826
 tonalitytools.ExtentIndicator (class in ab-
 jad.tools.tonalitytools.ExtentIndicator.ExtentIndicator),
 1809
 tonalitytools.InversionIndicator (class in ab-
 jad.tools.tonalitytools.InversionIndicator.InversionIndicator),
 1810
 tonalitytools.is_neighbor_note() (in module ab-
 jad.tools.tonalitytools.is_neighbor_note),
 1826
 tonalitytools.is_passing_tone() (in module ab-
 jad.tools.tonalitytools.is_passing_tone), 1827
 tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale()
 (in module ab-
 jad.tools.tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale),
 1827
 tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale()
 (in module ab-
 jad.tools.tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale),
 1827
 tonalitytools.make_first_n_notes_in_ascending_diatonic_scale()
 (in module ab-
 jad.tools.tonalitytools.make_first_n_notes_in_ascending_diatonic_scale),
 1828
 tonalitytools.Mode (class in ab-
 jad.tools.tonalitytools.Mode.Mode), 1812
 tonalitytools.OmissionIndicator (class in ab-
 jad.tools.tonalitytools.OmissionIndicator.OmissionIndicator),
 1813
 tonalitytools.QualityIndicator (class in ab-
 jad.tools.tonalitytools.QualityIndicator.QualityIndicator),
 1814
 tonalitytools.Scale (class in ab-
 jad.tools.tonalitytools.Scale.Scale), 1816
 tonalitytools.ScaleDegree (class in ab-
 jad.tools.tonalitytools.ScaleDegree.ScaleDegree),
 1819
 tonalitytools.SuspensionIndicator (class in ab-
 jad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator),
 1820
 tonalitytools.TonalFunction (class in ab-
 jad.tools.tonalitytools.TonalFunction.TonalFunction),
 1821
 tonalitytools.tonic (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark attribute), 404
 tools.AssignabilityError (class in ab-
 jad.tools.exceptiontools.AssignabilityError),
 1874
 tools.ClefError (class in ab-
 jad.tools.exceptiontools.ClefError), 1975
 tools.ContainmentError (class in ab-
 jad.tools.exceptiontools.ContainmentError),
 1976
 tools.ContextContainmentError (class in ab-
 jad.tools.exceptiontools.ContextContainmentError),
 1977
 tools.ContiguityError (class in ab-
 jad.tools.exceptiontools.ContiguityError),
 1978
 tools.CyclicNodeError (class in ab-
 jad.tools.exceptiontools.CyclicNodeError),
 1979
 tools.DurationError (class in ab-
 jad.tools.exceptiontools.DurationError), 1980
 tools.ExtraMarkError (class in ab-
 jad.tools.exceptiontools.ExtraMarkError),
 1981

tools.ExtraNamedComponentError (class in abjad.tools.exceptiontools.ExtraNamedComponentError), 1982	tools.MissingNoteHeadError (class in abjad.tools.exceptiontools.MissingNoteHeadError), 2001
tools.ExtraNoteHeadError (class in abjad.tools.exceptiontools.ExtraNoteHeadError), 1983	tools.MissingPitchError (class in abjad.tools.exceptiontools.MissingPitchError), 2002
tools.ExtraPitchError (class in abjad.tools.exceptiontools.ExtraPitchError), 1984	tools.MissingSpannerError (class in abjad.tools.exceptiontools.MissingSpannerError), 2003
tools.ExtraSpannerError (class in abjad.tools.exceptiontools.ExtraSpannerError), 1985	tools.MissingTempoError (class in abjad.tools.exceptiontools.MissingTempoError), 2004
tools.GraceContainerError (class in abjad.tools.exceptiontools.GraceContainerError), 1986	tools.MusicContentsError (class in abjad.tools.exceptiontools.MusicContentsError), 2005
tools.ImpreciseTempoError (class in abjad.tools.exceptiontools.ImpreciseTempoError), 1987	tools.NegativeDurationError (class in abjad.tools.exceptiontools.NegativeDurationError), 2006
tools.InputSpecificationError (class in abjad.tools.exceptiontools.InputSpecificationError), 1988	tools.NonbinaryTimeSignatureConversionError (class in abjad.tools.exceptiontools.NonbinaryTimeSignatureConversionError), 2007
tools.InstrumentError (class in abjad.tools.exceptiontools.InstrumentError), 1989	tools.NonbinaryTimeSignatureSuppressionError (class in abjad.tools.exceptiontools.NonbinaryTimeSignatureSuppressionError), 2008
tools.IntervalError (class in abjad.tools.exceptiontools.IntervalError), 1990	tools.NoteHeadError (class in abjad.tools.exceptiontools.NoteHeadError), 2009
tools.LilyPondParserError (class in abjad.tools.exceptiontools.LilyPondParserError), 1991	tools.OverfullContainerError (class in abjad.tools.exceptiontools.OverfullContainerError), 2010
tools.LineBreakError (class in abjad.tools.exceptiontools.LineBreakError), 1992	tools.ParallelError (class in abjad.tools.exceptiontools.ParallelError), 2011
tools.MarkError (class in abjad.tools.exceptiontools.MarkError), 1993	tools.PartitionError (class in abjad.tools.exceptiontools.PartitionError), 2012
tools.MeasureContiguityError (class in abjad.tools.exceptiontools.MeasureContiguityError), 1994	tools.PitchError (class in abjad.tools.exceptiontools.PitchError), 2013
tools.MeasureError (class in abjad.tools.exceptiontools.MeasureError), 1995	tools.SchemeParserFinishedException (class in abjad.tools.exceptiontools.SchemeParserFinishedException), 2014
tools.MissingComponentError (class in abjad.tools.exceptiontools.MissingComponentError), 1996	tools.SpacingError (class in abjad.tools.exceptiontools.SpacingError), 2015
tools.MissingInstrumentError (class in abjad.tools.exceptiontools.MissingInstrumentError), 1997	tools.SpannerError (class in abjad.tools.exceptiontools.SpannerError), 2016
tools.MissingMarkError (class in abjad.tools.exceptiontools.MissingMarkError), 1998	tools.SpannerPopulationError (class in abjad.tools.exceptiontools.SpannerPopulationError), 2017
tools.MissingMeasureError (class in abjad.tools.exceptiontools.MissingMeasureError), 1999	tools.StaffContainmentError (class in abjad.tools.exceptiontools.StaffContainmentError), 2018
tools.MissingNamedComponentError (class in abjad.tools.exceptiontools.MissingNamedComponentError), 2000	tools.TempoError (class in abjad.tools.exceptiontools.TempoError), 2019
	tools.TieChainError (class in abjad.tools.exceptiontools.TieChainError), 2020

(in module abjad.tools.tuplettools.set_denominator_of_tuplets_in_expr_to_at_least_1857),
 jad.tools.tuplettools.change_fixed_duration_tuplets_in_expr_to_at_least_1849
 tuplettools.Tuplet (class in abjad.tools.tuplettools.Tuplet.Tuplet), 1839
 (in module abjad.tools.tuplettools.change_tuplets_in_expr_to_fixed_duration_tuplets), 1850
 jad.tools.tuplettools.change_tuplets_in_expr_to_fixed_duration_tuplets(), 1850
 tuplettools.fix_contents_of_tuplets_in_expr() jad.tools.pitchtools.NumberedChromaticPitchClassColorMap.NumberedChromaticPitchClassColorMap, 1234
 (in module abjad.tools.tuplettools.fix_contents_of_tuplets_in_expr_to_twenty_four_tone_complete), 1850
 jad.tools.tuplettools.fix_contents_of_tuplets_in_expr_to_twenty_four_tone_complete() jad.tools.pitchtools.NumberedChromaticPitchClassColorMap.NumberedChromaticPitchClassColorMap, 1234
 tuplettools.FixedDurationTuplet (class in abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet), 1829
 jad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet(), 1829
 tuplettools.fuse_tuplets() (in module abjad.tools.tuplettools.fuse_tuplets), 1850
 U
 tuplettools.get_first_tuplet_in_improper_parentage_of_component() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet), 1140
 (in module abjad.tools.tuplettools.get_first_tuplet_in_improper_parentage_of_component()), 1851
 jad.tools.tuplettools.get_first_tuplet_in_improper_parentage_of_component() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet), 1150
 tuplettools.get_first_tuplet_in_proper_parentage_of_component() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet), 1155
 (in module abjad.tools.tuplettools.get_first_tuplet_in_proper_parentage_of_component()), 1852
 jad.tools.tuplettools.get_first_tuplet_in_proper_parentage_of_component() (abjad.tools.pitchtools.IntervalClassObjectSet.IntervalClassObjectSet), 1090
 tuplettools.leaf_to_tuplet_with_n_notes_of_equal_written_duration() (abjad.tools.pitchtools.IntervalObjectSet.IntervalObjectSet.IntervalObjectSet), 1098
 (in module abjad.tools.tuplettools.leaf_to_tuplet_with_n_notes_of_equal_written_duration()), 1852
 jad.tools.tuplettools.leaf_to_tuplet_with_n_notes_of_equal_written_duration() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet), 1162
 tuplettools.leaf_to_tuplet_with_proportions() union() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet), 1187
 (in module abjad.tools.tuplettools.leaf_to_tuplet_with_proportions()), 1853
 jad.tools.tuplettools.leaf_to_tuplet_with_proportions() union() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet), 1200
 tuplettools.make_tuplet_from_duration_and_proportions() union() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet), 1213
 (in module abjad.tools.tuplettools.make_tuplet_from_duration_and_proportions()), 1854
 jad.tools.tuplettools.make_tuplet_from_duration_and_proportions() union() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet), 1219
 tuplettools.make_tuplet_from_proportions_and_pair() union() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet), 1241
 (in module abjad.tools.tuplettools.make_tuplet_from_proportions_and_pair()), 1855
 jad.tools.tuplettools.make_tuplet_from_proportions_and_pair() union() (abjad.tools.pitchtools.ObjectSet.ObjectSet.ObjectSet), 1112
 tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet() union() (abjad.tools.pitchtools.PitchClassObjectSet.PitchClassObjectSet.PitchClassObjectSet), 1121
 (in module abjad.tools.tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet()), 1856
 jad.tools.tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet() union() (abjad.tools.pitchtools.PitchObjectSet.PitchObjectSet.PitchObjectSet), 1127
 tuplettools.remove_trivial_tuplets_in_expr() union() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass), 1805
 (in module abjad.tools.tuplettools.remove_trivial_tuplets_in_expr()), 1857
 jad.tools.tuplettools.remove_trivial_tuplets_in_expr() permits_per_minute (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark), 412
 tuplettools.scale_contents_of_tuplets_in_expr_by_multiplier() update() (abjad.tools.configurationtools.AbjadConfig.AbjadConfig.AbjadConfig), 1896
 (in module abjad.tools.tuplettools.scale_contents_of_tuplets_in_expr_by_multiplier()), 1857
 jad.tools.tuplettools.scale_contents_of_tuplets_in_expr_by_multiplier() update() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary), 1903
 tuplettools.set_denominator_of_tuplets_in_expr_to_at_least_1() update() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph), 1967
 (in module abjad.tools.tuplettools.set_denominator_of_tuplets_in_expr_to_at_least_1()), 1967

update() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector attribute), 1135

update() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector attribute), 1165

update() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector attribute), 1173

update() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector attribute), 1181

update() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector attribute), 1222

update() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector attribute), 1245

update() (abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector attribute), 1115

update() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval attribute), 1709

update() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1743

V

value (abjad.tools.marktools.Annotation.Annotation.Annotation attribute), 891

values() (abjad.tools.configurationtools.AbjadConfig.AbjadConfig attribute), 1896

values() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary attribute), 1903

values() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph attribute), 1967

values() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector attribute), 1135

values() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector attribute), 1165

values() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector attribute), 1173

values() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector attribute), 1181

values() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector attribute), 1222

values() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector attribute), 1245

values() (abjad.tools.pitchtools.ObjectVector.ObjectVector.ObjectVector attribute), 1115

values() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval attribute), 1709

values() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1743

version (abjad.tools.abjadbooktools.AbjadBookScript.AbjadBookScript attribute), 1887

version (abjad.tools.developerscripttools.AbjDevScript.AbjDevScript attribute), 1916

version (abjad.tools.developerscripttools.AbjGrepScript.AbjGrepScript attribute), 1918

version (abjad.tools.developerscripttools.BuildApiScript.BuildApiScript attribute), 1921

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1923

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1925

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1928

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1911

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1913

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1930

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1932

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1935

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1937

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1940

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1942

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1945

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1947

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1949

version (abjad.tools.developerscripttools.ChromaticIntervalClassVector.ChromaticIntervalClassVector attribute), 1952

verticalitytools.get_vertical_moment_at_offset_in_expr() (abjad.tools.verticalitytools.get_vertical_moment_at_offset_in_expr() attribute), 1861

verticalitytools.get_vertical_moment_starting_with_component() (abjad.tools.verticalitytools.get_vertical_moment_starting_with_component() attribute), 1862

verticalitytools.iterate_vertical_moments_in_expr() (abjad.tools.verticalitytools.iterate_vertical_moments_in_expr() attribute), 1863

verticalitytools.VerticalMoment (class in abjad.tools.verticalitytools.VerticalMoment.VerticalMoment), 1858

viewitems() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary attribute), 1904

viewitems() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph attribute), 1967

viewitems() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector attribute), 1135

viewitems() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector attribute), 1165

method), 1173
viewitems() (`abjad.tools.pitchtools.MelodicChromaticIntervalClassVector`), 1181
viewitems() (`abjad.tools.pitchtools.NamedChromaticPitchVector.NestedChromaticPitchClassNestedShorthandPitchCheck.DuplicateId` method), 1222
viewitems() (`abjad.tools.pitchtools.NumberedChromaticPitchClassVector`), 1245
viewitems() (`abjad.tools.pitchtools.ObjectVector.ObjectVectorOrObjectVector`), 1115
viewitems() (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary`), 1743
viewkeys() (`abjad.tools.datastructuretools.ImmutableDictionary`), 1904
viewkeys() (`abjad.tools.documentationtools.InheritanceGraph`), 1967
viewkeys() (`abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector`), 1136
viewkeys() (`abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector`), 1166
viewkeys() (`abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector`), 1173
viewkeys() (`abjad.tools.pitchtools.MelodicChromaticIntervalClassVector`), 1181
viewkeys() (`abjad.tools.pitchtools.NamedChromaticPitchVector.NestedChromaticPitchClassNestedShorthandPitchCheck.OverlapPitchCheck` method), 1222
viewkeys() (`abjad.tools.pitchtools.NumberedChromaticPitchClassVector`), 1245
viewkeys() (`abjad.tools.pitchtools.ObjectVector.ObjectVectorOrObjectVector`), 1115
viewkeys() (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary`), 1743
viewvalues() (`abjad.tools.datastructuretools.ImmutableDictionary`), 1904
viewvalues() (`abjad.tools.documentationtools.InheritanceGraph`), 1967
viewvalues() (`abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector`), 1136
viewvalues() (`abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector`), 1166
viewvalues() (`abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector`), 1173
viewvalues() (`abjad.tools.pitchtools.MelodicChromaticIntervalClassVector`), 1181
viewvalues() (`abjad.tools.pitchtools.NamedChromaticPitchVector.NestedChromaticPitchClassNestedShorthandPitchCheck.OverlapPitchCheck` method), 1222
viewvalues() (`abjad.tools.pitchtools.NumberedChromaticPitchClassVector`), 1245
viewvalues() (`abjad.tools.pitchtools.ObjectVector.ObjectVector.OrObjectVector` method), 1115
viewvalues() (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary`), 1743
violators() (`abjad.tools.wellformednesstools.BeamedQuarterNoteCheck`), 2082
violators() (`abjad.tools.wellformednesstools.Check`), 2082

W

- wait() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1971
- weight (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1050
- weight (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell attribute), 1053
- weight (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn attribute), 1055
- weight (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 1057
- wellformednesstools.BeamedQuarterNoteCheck (class in abjad.tools.wellformednesstools.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck), 2082
- wellformednesstools.Check (class in abjad.tools.wellformednesstools.Check.Check), 2080
- wellformednesstools.DiscontiguousSpannerCheck (class in abjad.tools.wellformednesstools.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck), 2083
- wellformednesstools.DuplicateIdCheck (class in abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck), 2085
- wellformednesstools.EmptyContainerCheck (class in abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck), 2086
- wellformednesstools.IntermarkedHairpinCheck (class in abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck), 2088
- wellformednesstools.list_checks() (in module abjad.tools.wellformednesstools.list_checks), 2104
- wellformednesstools.MisduratedMeasureCheck (class in abjad.tools.wellformednesstools.MisduratedMeasureCheck.MisduratedMeasureCheck), 2089
- wellformednesstools.MisfilledMeasureCheck (class in abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck), 2091
- wellformednesstools.MispitchedTieCheck (class in abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck), 2092
- wellformednesstools.MisrepresentedFlagCheck (class in abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck), 2094
- wellformednesstools.MissingParentCheck (class in abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck), 2095
- wellformednesstools.NestedMeasureCheck (class in abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck), 2097
- wellformednesstools.OverlappingBeamCheck (class in abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck), 2098
- wellformednesstools.OverlappingGlissandoCheck (class in abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck), 2100
- wellformednesstools.OverlappingOctavationCheck (class in abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck), 2101
- wellformednesstools.ShortHairpinCheck (class in abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck), 2103
- width (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1050
- width (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell attribute), 1053
- width (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn attribute), 1055
- width (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 1057
- width (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1426
- width (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1439
- withdraw() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1058
- write() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1971
- write_line() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1971
- written_duration (abjad.tools.beamtools.BeamSpanner.BeamSpanner.BeamSpanner attribute), 225
- written_duration (abjad.tools.beamtools.ComplexBeamSpanner.ComplexBeamSpanner attribute), 232
- written_duration (abjad.tools.beamtools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 240
- written_duration (abjad.tools.beamtools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 249
- written_duration (abjad.tools.beamtools.MultipartBeamSpanner.MultipartBeamSpanner attribute), 257
- written_duration (abjad.tools.chordtools.Chord.Chord.Chord attribute), 269
- written_duration (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 829
- written_duration (abjad.tools.notationtools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 1035
- written_duration (abjad.tools.notationtools.Note.Note.Note attribute), 1039
- written_duration (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest attribute), 1039

