auxjad Documentation

Release 0.2.3

Gilberto Agostinho

Feb 07, 2020

CONTENTS:

CHAPTER

ONE

THE *AUXJAD* PACKAGE

1.1 auxjad

Auxiliary functions and classes for Abjad 3.0. All classes and functions have a *__doc__* attribute with usage instructions.

Documentation is available at https://gilbertohasnofb.github.io/auxjad-docs/. A pdf version of the documentation is also available in the *docs* directory.

Bugs can be reported to https://github.com/gilbertohasnofb/auxjad/issues.

This library is published under the MIT License.

class auxjad.**LeafDynMaker**(*, *increase_monotonic: bool = None*, *forbidden_note_duration: Union[abjad.utilities.Duration.Duration, Tuple[int, int]] = None*, *forbidden_rest_duration: Union[abjad.utilities.Duration.Duration, Tuple[int, int]] = None*, *skips_instead_of_rests: bool = None*, *tag: abjad.system.Tag.Tag = None*, *use_multimeasure_rests: bool = None*)

An extension of abjad.LeafMaker which can also take optional lists of dynamics and articulations.

Usage is similar to LeafMaker:

```
>>> pitches = [0, 2, 4, 5, 7, 9]
>>> durations = [(1, 32), (2, 32), (3, 32), (4, 32), (5, 32), (6, 32)]
>>> dynamics = ['pp', 'p', 'mp', 'mf', 'f', 'ff']
>>> articulations = ['.', '>', '-', '_', '^', '+']
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches, durations, dynamics, articulations)
>>> staff = abjad.Staff(notes)
>>> abjad.f(staff)
\new Staff
{
    c'32
    \pp
    -\staccato
    d'16
    \p
    -\accent
    e'16.
    \mp
    -\tenuto
    f'8
    \mf
    -\portato
    g'8
```

(continues on next page)

(continued from previous page)

```
    \f
    -\marcato
    ~
    g'32
    a'8.
    \ff
    -\stopped
}
```

Tuple elements in `pitches` result in chords. None-valued elements in `pitches` result in rests:

```
>>> pitches = [5, None, (0, 2, 7)]
>>> durations = [(1, 4), (1, 8), (1, 16)]
>>> dynamics = ['p', None, 'f']
>>> articulations = ['staccato', None, 'tenuto']
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches, durations, dynamics, articulations)
>>> staff = abjad.Staff(notes)
>>> abjad.f(staff)
\new Staff
{
    f'4
    \p
    -\staccato
    r8
    <c' d' g'>16
    \f
    -\tenuto
}
```

Can omit repeated dynamics with the keyword argument `no_repeat`:

```
>>> pitches = [0, 2, 4, 5, 7, 9]
>>> durations = [(1, 32), (2, 32), (3, 32), (4, 32), (5, 32), (6, 32)]
>>> dynamics = ['pp', 'pp', 'mp', 'f', 'f', 'p']
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches,
...                        durations,
...                        dynamics,
...                        no_repeat=True,
...                        )
>>> staff = abjad.Staff(notes)
>>> abjad.f(staff)
\new Staff
{
    c'32
    \pp
    d'16
    e'16.
    \mp
    f'8
    \f
    g'8
    ~
    g'32
    a'8.
    \p
```

(continues on next page)

(continued from previous page)

```
}
```

The lengths of both `dynamics` and `articulations` can be shorter than the lengths of `pitches` and `durations` (whatever is the greatest):

```
>>> pitches = [0, 2, 4, 5, 7, 9]
>>> durations = (1, 4)
>>> dynamics = ['p', 'f', 'ff']
>>> articulations = ['.', '>']
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches, durations, dynamics, articulations)
>>> staff = abjad.Staff(notes)
>>> abjad.f(staff)
\new Staff
{
    c'4
    \p
    -\staccato
    d'4
    \f
    -\accent
    e'4
    \ff
    f'4
    g'4
    a'4
}
```

If the length of `articulations` is 1, it will apply to all elements. If the length of `dynamics` is 1, it will apply to the first element only:

```
>>> pitches = [0, 2, 4, 5, 7, 9]
>>> durations = (1, 4)
>>> dynamics = 'p'
>>> articulations = '.'
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches, durations, dynamics, articulations)
>>> staff = abjad.Staff(notes)
>>> abjad.f(staff)
\new Staff
{
    c'4
    \p
    -\staccato
    d'4
    -\staccato
    e'4
    -\staccato
    f'4
    -\staccato
    g'4
    -\staccato
    a'4
    -\staccato
}
```

Similarly to abjad's native classes, it accepts many types of elements in its input lists:

```
>>> pitches = [0,
...            "d'",
...            'E4',
...            abjad.Pitch(5),
...            abjad.Pitch("g'"),
...            abjad.Pitch("A4"),
...            ]
>>> durations = [1/32,
...              (2, 32),
...              0.09375,
...              abjad.Duration(0.125),
...              abjad.Duration((5, 32)),
...              abjad.Duration(6/32),
...              ]
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches, durations)
>>> staff = abjad.Staff(notes)
\new Staff
{
    c'32
    d'16
    e'16.
    f'8
    g'8
    ~
    g'32
    a'8.
}
```

auxjad.**container_comparator**(*container1: abjad.core.Container.Container*, *container2: abjad.core.Container.Container*, *include_indicators: bool = False*, *include_grace_notes: bool = False*) → bool

A comparator function returning True when two containers are identical and False when they are not.

When the pitches and effective durations of all leaves in both containers are identical, this function returns `True`:

```
>>> container1 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> auxjad.container_comparator(container1, container2)
True
```

Even if all leaves of both containers are identical in pitches and in written_duration, the function considers the effective duration so that situations like the one below do not yield a false positive:

```
>>> container1 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"\times 3/2 {c'4 d'4 e'4} f'4 <g' a'>2 r2")
>>> auxjad.container_comparator(container1, container2)
False
```

By default, this function ignores indicators, so the containers in the example below are understood to be identical:

```
>>> container1 = abjad.Staff(r"c'4\pp d'4 e'4-. f'4 <g' a'>2-> r2")
>>> container2 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> auxjad.container_comparator(container1, container2)
True
```

Setting the argument `include_indicators` to `True` forces the function to include indicators in its comparison. In that case, the containers in the example above are not considered identical any longer:

```
>>> container1 = abjad.Staff(r"c'4\pp d'4 e'4-. f'4 <g' a'>2-> r2")
>>> container2 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> auxjad.container_comparator(container1,
...                             container2,
...                             include_indicators=True,
...                             )
True
```

By default, this function ignores grace notes, so the containers in the example below are understood to be identical:

```
>>> container1 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"c'4 \grace{c''4} d'4 e'4 f'4 <g' a'>2 r2")
>>> auxjad.container_comparator(container1, container2)
True
```

Setting the argument `include_grace_notes` to `True` forces the function to include grace notes in its comparison. In that case, the containers in the example above are not considered identical any longer:

```
>>> container1 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"c'4 \grace{c''4} d'4 e'4 f'4 <g' a'>2 r2")
>>> auxjad.container_comparator(container1,
...                             container2,
...                             include_grace_notes=True,
...                             )
False
```

When the argument `include_grace_notes` is set to `True`, the function will consider not only a grace note is attached to a given leaf, but also wether the contents of the grace containers are the identical:

```
>>> container1 = abjad.Staff(r"c'4 \grace{c''4} d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"c'4 \grace{c''8} d'4 e'4 f'4 <g' a'>2 r2")
>>> auxjad.container_comparator(container1,
...                             container2,
...                             include_grace_notes=True,
...                             )
False
```

auxjad.**time_signature_remover**(*container: abjad.core.Container.Container*) → abjad.core.Container.Container

A function which removes all unecessary time signatures. It removes consecutive effective time signatures, even if separated by any number of bars with no time signature.

When two consecutive bars have identical time signatures, the second one is removed:

```
>>> staff = abjad.Staff(r"c'4 d'8 | c'4 d'8")
>>> abjad.attach(abjad.TimeSignature((3, 8)), staff[0])
>>> abjad.attach(abjad.TimeSignature((3, 8)), staff[2])
>>> abjad.f(staff)
\new Staff
{
    \time 3/8
    c'4
    d'8
    \time 3/8
```

(continues on next page)

(continued from previous page)

```
    c'4
    d'8
}
>>> staff = auxjad.time_signature_remover(staff)
>>> abjad.f(staff)
\new Staff
{
    \time 3/8
    c'4
    d'8
    c'4
    d'8
}
```

The function also removes time signatures that are separated by an arbitrary number of bars without one:

```
>>> staff = abjad.Staff(r"c'4 d'8 e'4. c'4 d'8")
>>> abjad.attach(abjad.TimeSignature((3, 8)), staff[0])
>>> abjad.attach(abjad.TimeSignature((3, 8)), staff[3])
>>> abjad.f(staff)
\new Staff
{
    \time 3/8
    c'4
    d'8
    e'4.
    \time 3/8
    c'4
    d'8
}
>>> staff = auxjad.time_signature_remover(staff)
>>> abjad.f(staff)
\new Staff
{
    \time 3/8
    c'4
    d'8
    e'4.
    c'4
    d'8
}
```

The input container can also handle subcontainers, including cases in which the time signatures are attached to leaves of subcontainers:

```
>>> staff = abjad.Staff([abjad.Note("c'2"),
...                      abjad.Chord("<d' f'>2"),
...                      abjad.Tuplet((2, 3), "g2 a2 b2"),
...                      ])
>>> abjad.attach(abjad.TimeSignature((2, 2)), staff[0])
>>> abjad.attach(abjad.TimeSignature((2, 2)), staff[2][0])
>>> abjad.f(staff)
\new Staff
{
    \time 2/2
    c'2
```

(continues on next page)

(continued from previous page)

```
    <d' f'>2
    \times 2/3 {
        \time 2/2
        g2
        a2
        b2
    }
}
>>> staff = auxjad.time_signature_remover(staff)
>>> abjad.f(staff)
\new Staff
{
    \time 2/2
    c'2
    <d' f'>2
    \times 2/3 {
        g2
        a2
        b2
    }
}
```

PYTHON MODULE INDEX

a

INDEX