auxjad Documentation

Release 0.2.3

Gilberto Agostinho

Feb 08, 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.`**`remove_repeated_dynamics`**(*container: abjad.core.Container.Container*, *ignore_hairpins: bool = False*, *reset_after_rests: bool = False*) → abjad.core.Container.Container

A function which removes all consecutive repeated dynamics. It removes consecutive effective dynamics, even if separated by any number of notes without one. It resets its memory of what was the previous dynamic every time it finds a hairpin, since notation such as "c'4f> c'4f>" is quite common; this behaviour can be toggled off using the ignore_hairpins keyword. By default, it remembers the previous dynamic even with notes separated by rests; this can be toggled off using reset_after_rests=True. To set a maximum length of silence after which dynamics are restated, set reset_after_rests to a duration using abjad.Duration() or any other duration format accepted by Abjad.

When two consecutive leaves have identical dynamics, the second one is removed:

```
>>> staff = abjad.Staff(r"c'4\pp d'8\pp | c'4\f d'8\f")
>>> abjad.f(staff)
\new Staff
```

(continues on next page)

(continued from previous page)

```
{
    c'4
    \pp
    d'8
    \pp
    c'4
    \f
    d'8
    \f
}
>>> staff = auxjad.remove_repeated_dynamics(staff)
>>> abjad.f(staff)
\new Staff
{
    c'4
    \pp
    d'8
    c'4
    \f
    d'8
}
```

The function also removes dynamics that are separated by an arbitrary number of leaves without dynamics:

```
>>> staff = abjad.Staff(r"c'4\p d'8 | e'4.\p | c'4\p d'8\f")
>>> abjad.f(staff)
\new Staff
{
    c'4
    \p
    d'8
    e'4.
    \p
    c'4
    \p
    d'8
    \f
}
>>> staff = auxjad.remove_repeated_dynamics(staff)
>>> abjad.f(staff)
\new Staff
{
    c'4
    \p
    d'8
    e'4.
    c'4
    d'8
    \f
}
```

The input container can also handle subcontainers:

```
>>> staff = abjad.Staff([abjad.Note("c'2"),
...                      abjad.Chord("<d' f'>2"),
...                      abjad.Tuplet((2, 3), "g2 a2 b2"),
...                      ])
```

(continues on next page)

(continued from previous page)

```
>>> abjad.attach(abjad.Dynamic('ppp'), staff[0])
>>> abjad.attach(abjad.Dynamic('ppp'), staff[1])
>>> abjad.attach(abjad.Dynamic('ppp'), staff[2][0])
>>> abjad.f(staff)
\new Staff
{
    c'2
    \ppp
    <d' f'>2
    \ppp
    \times 2/3 {
        g2
        \ppp
        a2
        b2
    }
}
>>> staff = auxjad.remove_repeated_dynamics(staff)
>>> abjad.f(staff)
\new Staff
{
    c'2
    \ppp
    <d' f'>2
    \times 2/3 {
        g2
        a2
        b2
    }
}
```

By default, repeated dynamics with hairpins in between are not removed, but consecutive ones will.

```
>>> staff = abjad.Staff(r"c'4\pp\< d'8\f\> | c'4\f d'8\f")
>>> abjad.f(staff)
\new Staff
{
    c'4
    \pp
    \<
    d'8
    \f
    \>
    c'4
    \f
    d'8
    \f
}
>>> staff = auxjad.remove_repeated_dynamics(staff)
>>> abjad.f(staff)
\new Staff
{
    c'4
    \pp
    \<
    d'8
    \f
```

(continues on next page)

(continued from previous page)

```
    \>
    c'4
    \f
    d'8
}
```

To override the previous behaviour, set ignore_hairpins to True and hairpins will be ignored.

```
>>> staff = abjad.Staff(r"c'4\pp\< d'8\f\> | c'4\f d'8\f")
>>> abjad.f(staff)
\new Staff
{
    c'4
    \pp
    \<
    d'8
    \f
    \>
    c'4
    \f
    d'8
    \f
}
>>> staff = auxjad.remove_repeated_dynamics(staff,
...                                         ignore_hairpins=True,
...                                         )
>>> abjad.f(staff)
\new Staff
{
    c'4
    \pp
    \<
    d'8
    \f
    \>
    c'4
    d'8
}
```

By default, rests are treated just like any other leaf and thus notes with an identical dynamic separated by an arbitrary number of rests will be considered as repeated and the second dynamic will be removed.

```
>>> staff = abjad.Staff(r"c'4\pp r2. | c'4\pp")
>>> staff = auxjad.remove_repeated_dynamics(staff)
>>> abjad.f(staff)
\new Staff
{
    c'4
    \pp
    r2.
    c'4
}
```

To override the previous behaviour, set reset_after_rests to True and dynamics will always be restated after a rest.

```
>>> staff = abjad.Staff(r"c'4\pp r2. | c'4\pp")
>>> staff = auxjad.remove_repeated_dynamics(staff,
...                                         reset_after_rests=True,
...                                         )
>>> abjad.f(staff)
\new Staff
{
    c'4
    \pp
    r2.
    c'4
    \pp
}
```

The argument reset_after_rests takes not only boolean values but also duration (abjad.Duration, tuple, float, etc.). This sets the maximum length of rests before which identical dynamics are restated. If the total length of rests falls below that value, then repeated dynamics are removed.

In the case below, a rest of r2. is shorter than a duration of (4, 4), so the repeated dynamic is removed.

```
>>> staff = abjad.Staff(r"c'4\pp r2. | c'4\pp")
>>> staff = auxjad.remove_repeated_dynamics(staff,
...                                         reset_after_rests=(4, 4),
...                                         )
>>> abjad.f(staff)
\new Staff
{
    c'4
    \pp
    r2.
    c'4
}
```

But setting the duration to 2/4 forces the dynamic to be restated.

```
>>> staff = abjad.Staff(r"c'4\pp r2. | c'4\pp")
>>> staff = auxjad.remove_repeated_dynamics(staff,
...                                         reset_after_rests=2/4,
...                                         )
>>> abjad.f(staff)
\new Staff
{
    c'4
    \pp
    r2.
    c'4
    \pp
}
```

The function also handles measure rests with reset_after_rests.

```
>>> staff = abjad.Staff(r"c'4\pp r2. | R1 | c'4\pp")
>>> staff = auxjad.remove_repeated_dynamics(
...     staff,
...     reset_after_rests=abjad.Duration(4, 4),
... )
>>> abjad.f(staff)
```

(continues on next page)

(continued from previous page)

```
\new Staff
{
    c'4
    \pp
    r2.
    R1
    c'4
    \pp
}
```

auxjad.**remove_repeated_time_signature**(*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
    c'4
    d'8
}
>>> staff = auxjad.remove_repeated_time_signature(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
}
```

(continues on next page)

(continued from previous page)

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

`auxjad.`**`simplified_time_signature_ratio`**(*ratio: tuple*, ***, *min_denominator: int = 4*) → tuple

A function simplifies the ratio of a given time signature respecting a minimum denominator value. Input is a tuple of two integers.

By default, the function simplifies the ratio of numerator/denominator using a minimum denominator value of 4 (that is, the denominator will not get smaller than 4). In the case below, (2, 4) is the simplest representaion of the ratio (4, 8) with a denominator equal to or larger than 4.

```
>>> ratio = auxjad.simplified_time_signature_ratio((4, 8))
>>> time_signature = abjad.TimeSignature(ratio)
>>> format(time_signature)
abjad.TimeSignature((2, 4))
```

If a ratio cannot be simplified at all, the function returns the original values.

```
>>> ratio = auxjad.simplified_time_signature_ratio((7, 8))
>>> time_signature = abjad.TimeSignature(ratio)
>>> format(time_signature)
abjad.TimeSignature((7, 8))
```

The min_denominator can be set to values other than 4. If set to 2, the simplest representaion of the ratio (4, 8) becomes (1, 2).

```
>>> ratio = auxjad.simplified_time_signature_ratio((4, 8),
...                                                  min_denominator=2
...                                                  )
>>> time_signature = abjad.TimeSignature(ratio)
>>> format(time_signature)
abjad.TimeSignature((1, 2))
```

PYTHON MODULE INDEX

INDEX