
Wavematic

Micael Jarniac

Aug 03, 2021

CONTENTS:

1	Wavematic	1
1.1	wavematic package	1
2	Indices and tables	5
	Python Module Index	7
	Index	9

WAVEMATIC

1.1 wavematic package

1.1.1 Module contents

Simple wave generator

exception `wavematic.MissingTimeAxis`

Bases: `Exception`

class `wavematic.Noise`(*ta: Optional[wavematic.TimeAxis] = None, amp: float = 0.0, seed: int = 0, name: str = 'Noise'*)

Bases: `wavematic.wave.Signal`

Generates noise.

Parameters

- **ta** – Time axis.
- **amp** – Amplitude.
- **seed** – Noise seed. Using the same seed generates the same noise signal.
- **name** – Name to give to the noise signal.

__abstractmethods__ = `frozenset({})`

__init__(*ta: Optional[wavematic.TimeAxis] = None, amp: float = 0.0, seed: int = 0, name: str = 'Noise'*)

__repr__() → str

Return repr(self).

get(*ta: Optional[wavematic.TimeAxis] = None*) → `pandas.core.series.Series`

Generate the noise signal.

Parameters **ta** – Time axis to use.

Returns Generated noise signal.

lacunarity = 2.0

octaves = 20

persistence = 5.0

repeat = 1024

class wavematic.**TimeAxis**(duration: float, rate: float, start: float = 0.0)

Bases: object

Generates a time axis.

Parameters

- **duration** – Length, in units of time. Non-negative.
- **rate** – Sampling rate (points per unit of time). Non-negative.
- **start** – Initial time.

__init__(duration: float, rate: float, start: float = 0.0)

__repr__() → str
Return repr(self).

get() → pandas.core.series.Series
Generate the time axis.

Returns Generated time axis.

class wavematic.**Wave**(ta: Optional[wavematic.TimeAxis] = None, freq: float = 0.0, amp: float = 0.0, phase: float = 0.0, disp: float = 0.0, kind: str = 'sine', name: Optional[str] = None, **kwargs)

Bases: wavematic.wave.Signal

Generates a waveform.

Parameters

- **ta** – Time axis.
- **freq** – Frequency (shouldn't be higher than half of the “sampling rate” on the time axis).
- **amp** – Amplitude.
- **phase** – Phase, in Pi (between 0.0 and 2.0).
- **disp** – Displacement.
- **kind** – The kind of wave to generate. Can be “sine” (default), “square” or “sawtooth”.
- **name** – The name to give to the wave signal.
- ****kwargs** – Extra arguments to be sent to the generator function. If kind=“square”, duty can be given. If kind=“sawtooth”, width can be given.

__abstractmethods__ = frozenset({})

__init__(ta: Optional[wavematic.TimeAxis] = None, freq: float = 0.0, amp: float = 0.0, phase: float = 0.0, disp: float = 0.0, kind: str = 'sine', name: Optional[str] = None, **kwargs)

__repr__() → str
Return repr(self).

copy() → wavematic.Wave
Create a shallow copy of itself.

get(ta: Optional[wavematic.TimeAxis] = None) → pandas.core.series.Series
Generate the wave signal.

Parameters **ta** – Time axis to use. If not provided, *self.ta* will be used.

Returns Generated wave signal.

```
class wavematic.Wavematic(ta: Optional[wavematic.TimeAxis] = None, name: Optional[str] = None)
    Bases: wavematic.wave.Signal
```

Combines multiple signals.

Parameters

- **ta** – Base time axis.
- **name** – The name to give to the resulting signal.

```
__abstractmethods__ = frozenset({})
```

```
__add__(other: Any) → wavematic.Wavematic
    Generate new Wavematic instance with added signal.
```

```
__annotations__ = {'force_self_ta': <class 'bool'>}
```

```
__iadd__(other: Any) → wavematic.Wavematic
    Shortcut to add a signal.
```

```
__init__(ta: Optional[wavematic.TimeAxis] = None, name: Optional[str] = None)
```

```
__repr__() → str
    Return repr(self).
```

```
add_signal(sig: wavematic.wave.Signal) → wavematic.Wavematic
    Add a signal.
```

Parameters **sig** – The signal to add.

Returns Reference to self.

```
all_signals(ta: Optional[wavematic.TimeAxis] = None) → pandas.core.frame.DataFrame
    Group all signals.
```

```
copy() → wavematic.Wavematic
    Create a deep copy of itself.
```

```
force_self_ta: bool = False
```

```
get(ta: Optional[wavematic.TimeAxis] = None) → pandas.core.series.Series
    Generate the signal resulting from the addition of contained signals.
```


INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

W

wavemantic, 1

Symbols

__abstractmethods__ (wavematic.Noise attribute), 1
 __abstractmethods__ (wavematic.Wave attribute), 2
 __abstractmethods__ (wavematic.Wavematic attribute), 3
 __add__() (wavematic.Wavematic method), 3
 __annotations__ (wavematic.Wavematic attribute), 3
 __iadd__() (wavematic.Wavematic method), 3
 __init__() (wavematic.Noise method), 1
 __init__() (wavematic.TimeAxis method), 2
 __init__() (wavematic.Wave method), 2
 __init__() (wavematic.Wavematic method), 3
 __repr__() (wavematic.Noise method), 1
 __repr__() (wavematic.TimeAxis method), 2
 __repr__() (wavematic.Wave method), 2
 __repr__() (wavematic.Wavematic method), 3

A

add_signal() (wavematic.Wavematic method), 3
 all_signals() (wavematic.Wavematic method), 3

C

copy() (wavematic.Wave method), 2
 copy() (wavematic.Wavematic method), 3

F

force_self_ta (wavematic.Wavematic attribute), 3

G

get() (wavematic.Noise method), 1
 get() (wavematic.TimeAxis method), 2
 get() (wavematic.Wave method), 2
 get() (wavematic.Wavematic method), 3

L

lacunarity (wavematic.Noise attribute), 1

M

MissingTimeAxis, 1
 module
 wavematic, 1

N

Noise (class in wavematic), 1

O

octaves (wavematic.Noise attribute), 1

P

persistence (wavematic.Noise attribute), 1

R

repeat (wavematic.Noise attribute), 1

T

TimeAxis (class in wavematic), 1

W

Wave (class in wavematic), 2
 wavematic
 module, 1
 Wavematic (class in wavematic), 2