SigFeat Documentation

Release 0.2a0

Siegfried Gündert

Aug 21, 2019

Contents

A Signal Feature Extraction framework for Python. Focused on audio signals.

Follows a clear design:

- **Source**
- **Feature**
- **Extractor**
- **Sink**

With **Parameters**, **Metadata** handling and **Preprocessing** capability. All needed information can be written to the Sink in a clear structure.

Contents:

CHAPTER 1

Examples

CHAPTER 2

Highlevel API

On top of the base API this library already implements some useful classes:

- An `ArraySource`: Source for numpy arrays with slicing functionaliy. And a `SoundFileSource`: Source for sound files using pysoundfile.
- Commonly known spectral, temporal and other Features (e.g. `RootMeanSquare` or `SpectralFlatness` etc.)
- A `DefaultDictSink`: Sink receiving results in dictionary.

2.1 Source

class sigfeat.source.array.**ArraySource**(*array*, *samplerate*, *name=''*, ***parameters*)

Source class for iterable arrays.

Parameters

array [ndarray] Expects an iterable array with .shape tuple.

samplerate [int]

name [str]

blocksize [int]

overlap [int]

Attributes

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
generate(self)	Returns generator that yields blocks out of the array.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.

get_class_parameters

generate(*self*)
Returns generator that yields blocks out of the array.

class sigfeat.source.soundfile.**SoundFileSource**(*sf=None*, ***parameters*)
Source generating data from SoundFiles.

The parameters are for the SoundFile.blocks() method used for the blocks method of this Source class.

Parameters

sf [SoundFile instance or str]

blocksize [int]

overlap [int]

frames [int] The number of frames to yield from soundfile. If frames < 1, the file is read until the end.

fill_value [scalar] The value last block will filled up, if it is shorter tha blocksize.

dtype [{'float64', 'float32', 'int32', 'int16'}, optional] See soundfile.SoundFile.read().

always_2d [bool] Indicates wether all blocks are at least 2d numpy arrays.

Attributes

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
generate(self)	Returns generator that yields blocks from the SoundFile.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.

get_class_parameters

generate (*self*)
Returns generator that yields blocks from the SoundFile.

2.2 Feature

class sigfeat.feature.**Index**(*name=None*, *requirements=None*, ***parameters*)
Index of source.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, result)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

process (*self*, *data*, *result*)
Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

class sigfeat.feature.**SpectralFlux**(*name=None*, *requirements=None*, ***parameters*)

Flux of AbsRfft.

$$D_m[k] = \frac{|X_m[k]|}{\max(|X_m[k]|)} - \frac{|X_{m-1}[k]|}{\max(|X_{m-1}[k]|)}$$

$$SFX_m = \frac{1}{2}\sum_k D[k]|D[k]|$$

Parameters

axis [int] Axis along the flux will be calculated, default=0.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, featuredata)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

on_start (*self*, *source*, *featureset*, *sink*)
Override this method if your feature needs some initialization.

The Extractor will give you source, featureset and sink.

Parameters

source [source]

featureset [OrderedDict of features]

sink [Sink]

process (*self*, *data*, *featuredata*)
Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)
Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**SpectralRolloff** (*name=None*, *requirements=None*, ***parameters*)
Rolloff from AbsRfft.

The spectral rolloff is the frequency where the kappa percentage of energy is below and the 1-kappa percentage of energy is above.

TODO formula

Parameters

kappa [scalar {0...1}] Default 0.95

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, result)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

on_start (*self*, *source*, *featureset*, *sink*)

Override this method if your feature needs some initialization.

The Extractor will give you source, featureset and sink.

Parameters

source [source]

featureset [OrderedDict of features]

sink [Sink]

process (*self*, *data*, *result*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires(*self*)
Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**SpectralCentroid**(*name=None*, *requirements=None*, ***parameters*)
Centroid of AbsRfft.

$$SC = \frac{\sum_k f[k]|X[k]|}{\frac{1}{K}\sum_k |X[k]|}$$

Parameters

axis [int] Axis along the centroid will be calculated, default=0.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, resd)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

centroid	
get_class_parameters	

on_start(*self*, *source*, *featureset*, *sink*)
Override this method if your feature needs some initialization.

The Extractor will give you source, featureset and sink.

Parameters

source [source]

featureset [OrderedDict of features]

sink [Sink]

process(*self*, *data*, *resd*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires(*self*)

Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**SpectralSpread**(*name=None*, *requirements=None*, ***parameters*)

Spread of AbsRfft.

$$SSP_m = \frac{\sum_k (f[k]SC_m[k])^2 |X_m[k]|}{\sum_k |X_m[k]|}$$

Parameters

axis [int] Axis along the centroid will be calculated, default=0.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.

Continued on next page

Table 7 – continued from previous page

fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, resd)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

centroid	
get_class_parameters	
spread	

process (*self*, *data*, *resd*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)

Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**SpectralSkewness** (*name=None*, *requirements=None*, ***parameters*)

Skewness of AbsRfft.

$$SSK_m = \frac{\sum_k (f[k] - SC_m[k])^3 |X_m[k]|}{\sqrt{SSP_m}^3 \sum_k |X_m[k]|}$$

Parameters

axis [int] Axis along the centroid will be calculated, default=0.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, resd)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

centroid	
get_class_parameters	
skewness	

process (*self*, *data*, *resd*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)

Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**SpectralKurtosis**(*name=None*, *requirements=None*, ***parameters*)

Kurtosis of AbsRfft.

$$SK_m = \frac{\sum_k (f[k] - SC_m[k])^4 |X_m[k]|}{SSP_m^2 \sum_k |X_m[k]|}$$

Parameters

axis [int] Axis along the centroid will be calculated, default=0.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, resd)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

centroid	
get_class_parameters	
kurtosis	

process(*self*, *data*, *resd*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires(*self*)

Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**SpectralFlatness**(*name=None*, *requirements=None*, ***parameters*)

Flatness of AbsRfft.

$$SF_m = \frac{(\Pi_k |X_m[k]|)^{1/K}}{\frac{1}{K}\sum_k |X_m[k]|}$$

Parameters

axis [int] Axis along the flatness will be calculated, default=0.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, featuredata)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.

Continued on next page

Table 10 – continued from previous page

`unroll_parameters`(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
`validate_name`(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters

process (*self*, *data*, *featuredata*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)

Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class `sigfeat.feature.`**SpectralCrestFactor** (*name=None*, *requirements=None*, ***parameters*)

Crest Factor of AbsRfft

$$SCF_m = \frac{\max(|X_m|)}{X_{m,RMS}}$$

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

`add_metadata`(self, name, value)	Appends the key value pair to metadata list.
`dependencies`(self)	Yields all dependencies of this feature.
`extend_metadata`(self, mdata)	Extends the metadata with the given list of key value pairs.
`featureset`(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
`fetch_metadata_as_attrs`(self)	Sets metadata as attributes of self.

Continued on next page

Table 11 – continued from previous page

gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, result)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters

process (*self*, *data*, *result*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)

Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**SpectralSlope** (*name=None*, *requirements=None*, ***parameters*)

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, features, sink)	Override this method if your feature needs some initialization.
process(self, data, resd)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

on_start (*self*, *source*, *features*, *sink*)

Override this method if your feature needs some initialization.

The Extractor will give you source, featureset and sink.

Parameters

source [source]

featureset [OrderedDict of features]

sink [Sink]

process (*self*, *data*, *resd*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires(*self*)
Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**MFCC**(*name=None*, *requirements=None*, ***parameters*)

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, resd)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

process(*self*, *data*, *resd*)
Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)

Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**RootMeanSquare**(*name=None*, *requirements=None*, ***parameters*)

Root Mean Square (RMS) from Source data.

$$RMS_m = \sqrt{\frac{1}{N}\sum_n x_m[n]^2}$$

Parameters

axis [int] Axis along which the feature is computed.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, result)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

process (*self*, *data*, *result*)
Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)
Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**CrestFactor** (*name=None*, *requirements=None*, ***parameters*)
Crest Factor of Source data.

Depends on Peak and RootMeanSquare features.

$$CF_m = \frac{\max(|x_m|)}{x_{m,RMS}}$$

Parameters

axis: int

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.

Continued on next page

Table 15 – continued from previous page

on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, resd)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

process (*self*, *data*, *resd*)
Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)
Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**ZeroCrossingRate** (*name=None*, *requirements=None*, ***parameters*)
Zero Crossings Rate of Source data.

$$ZCR_m = \frac{f_s}{N}\lfloor 0.5 + \frac{1}{2}\sum_n |\mathrm{sgn}(x_m[n+1]) - \mathrm{sgn}(x_m[n])|\rfloor$$

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.

Continued on next page

Table 16 – continued from previous page

dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, result)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

on_start (*self*, *source*, *featureset*, *sink*)

Override this method if your feature needs some initialization.

The Extractor will give you source, featureset and sink.

Parameters

source [source]

featureset [OrderedDict of features]

sink [Sink]

process (*self*, *data*, *result*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

class sigfeat.feature.**Peak**(*name=None*, *requirements=None*, ***parameters*)

Peak of AbsSignal from Source data.

$$P_m = \max(|x_m|)$$

Parameters

axis [int] Axis along which the feature is computed.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, result)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

process(*self*, *data*, *result*)

Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)
Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

class sigfeat.feature.**Delta** (*feature*, ***parameters*)
Returns a diffenetiated version of the given feature.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, resultd)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

process (*self*, *data*, *resultd*)
Override this method returning process results.

The data from the source will be in `data` (usually `data = block, index, ...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)

Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

2.3 Extractor

This Module contains the central Extractor class.

The Extractor consumes features instances and is used to extract the Features from given sources into given Sink.

class sigfeat.extractor.**Extractor**(**features*, *autoinst=True*)

Feature Extractor

Parameters

***features** [Feature instances] Provide multple feature instances. If you provide a Feature required by other features, please place it before the others.

autoinst [bool] Autoinstantiate required feature Classes if no instance exists in featureset. If False you will get errors with a hint which feature instance you need provide as well.

Methods

extract(self, source[, sink])	Extracts features from given source into given sink.
get_features_parameters_and_metadata(self, ...)	Returns dict with parameters and metadata from self.featureset.
get_parameters_and_metadata(obj)	Returns dict with parameters and metadata from given `obj`.
reset(self)	Resets the states of features.

extract (*self*, *source*, *sink=None*)

Extracts features from given source into given sink.

Parameters

source [Source instance]

sink [Sink instance]

Returns

res [Sink] The sink with processed data and metadata.

get_features_parameters_and_metadata(*self*, *hidden=False*)
Returns dict with parameters and metadata from self.featureset.

static get_parameters_and_metadata(*obj*)
Returns dict with parameters and metadata from given `obj`.

reset(*self*)
Resets the states of features.

If a new source shall be processed this may be usefull or needed.

2.4 Sink

class sigfeat.sink.default.**DefaultDictSink**
The receive_append method appends input to 'results' defaultdict(list)

Methods

clear()	
copy()	
fromkeys(iterable[, value])	Create a new dictionary with keys from iterable and values set to value.
get(self, key[, default])	Return the value for key if key is in the dictionary, else default.
items()	
keys()	
pop()	If key is not found, d is returned if given, otherwise KeyError is raised
popitem()	2-tuple; but raise KeyError if D is empty.
receive(self, datad)	Updates the dict with given `datad` dictionary.
receive_append(self, resultd)	Appends given `resultd` dict to list fields in this dict.
setdefault(self, key[, default])	Insert key with a value of default if key is not in the dictionary.
update()	If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
values()	

receive(*self*, *datad*)
Updates the dict with given `datad` dictionary.

receive_append(*self*, *resultd*)
Appends given `resultd` dict to list fields in this dict.

2.5 Preprocess

class sigfeat.preprocess.mix.**MeanMix**(*source*, ***parameters*)

Averages multi channel source e.g stereo to mono mix.

Parameters

axis [int]

channels [int] Number of channels will be generated by this preprocess.

Attributes

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
generate(self)	Override this method.
process(self, data)	Override this method.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.

get_class_parameters	

process(*self*, *data*)

Override this method.

Parameters

data [data from source]

class sigfeat.preprocess.mix.**SumMix**(*source*, ***parameters*)

Sums up multi channel source.

Parameters

axis [int]

channels [int] Number of channels will be generated by this preprocess.

Attributes

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.

Continued on next page

Table 22 – continued from previous page

extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
generate(self)	Override this method.
process(self, data)	Override this method.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.

get_class_parameters	

process (*self*, *data*)
Override this method.

Parameters

data [data from source]

CHAPTER 3

Base level API

Contains the abstract base classes.

3.1 Source

Implements abstract Source base class

class sigfeat.base.source.**Source**(***parameters*)

Base Source Class.

The parameters are mandatory for all kinds of sources. Every source must implement a `.generate()` method returning a generator that yields the signal blocks. Source must have samplerate and channels as metadata.

Parameters

blocksize [int]

overlap [int]

samplerate [scalar]

channels [int]

Attributes

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.

Continued on next page

Table 1 – continued from previous page

fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
generate(self)	Override this method.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.

get_class_parameters	

generate(*self*)

Override this method. It must yield `data`.

Usually `data = (block, index)`

3.2 Feature

This module implements the abstract base Feature class.

Feature is subclassed for implementing Signal Features.

TODO: What happens if i have a hidden feature but add a instance of tha same that is not hidden? Mus be solved in the FeatureSet! TODO: Handle labels for multidimensional feature output.

class sigfeat.base.feature.**Feature**(*name=None*, *requirements=None*, ***parameters*)

Abstract Feature Base Class

Parameters

name [str]

requirements [Feature instances iterable] To override those returned by self.requires().

Notes

At least the *process()* method must be overridden. If your implemented feature depends on the results of another feature, you must override the *requires()* method returning an iterable e.g. list of feature instances.

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.

Continued on next page

Table 2 – continued from previous page

fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, result)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters	

dependencies (*self*)
Yields all dependencies of this feature.

featureset (*self*, *new=False*, *autoinst=False*, *err_missing=True*)
Returns an ordered dict of all features unique in name.

The dict is ordered by the dependency tree order.

Parameters

new [boolean] All features will be reinitialized.

Returns

featdict [OrderedDict] Keys are `fid` and values are feature instances.

fid
Returns the feature identifying tuple.

gen_dependencies_instances (*self*, *autoinst=False*, *err_missing=True*)
Checks deps for being instance or class and yields instances.

hidden
Returns whether the feature is hidden or not.

hide (*self*, *b=True*)
Hide the feature.

new (*self*)
Returns new initial feature instance with same parameters.

on_finished (*self*, *source*, *featureset*, *sink*)
Override this method to be run after extraction.

Parameters

source [source]

featureset [OrderedDict of features]

sink [Sink]

on_start (*self*, *source*, *featureset*, *sink*)
Override this method if your feature needs some initialization.

The Extractor will give you source, featureset and sink.

Parameters

source [source]

featureset [OrderedDict of features]

sink [Sink]

process (*self*, *data*, *result*)
Override this method returning process results.

The data from the source will be in `data` (usually `data = block,index,...,`).

The required results from other features will be in the result argument (dict like *Result()*).

Parameters

data [block, index] Or the data generated from your source.

result [Result] The results from other features of current data. If you provide your required features in the requires() method, you will be able to access the results from those features.

Returns

res [e.g. scalar, ndarray or other custom types.] This is the feature result for the current data (block).

requires (*self*)
Override this method if your feature depends on other features.

You can return Feature classes and Feature instances you need for your feature. Yielding is also allowed.

validate_name (*self*)
Checks for uniqueness of feature name in all dependent features.

class sigfeat.base.feature.**HiddenFeature** (*name=None*, *requirements=None*, ***parameters*)

Attributes

fid Returns the feature identifying tuple.

hidden Returns whether the feature is hidden or not.

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
dependencies(self)	Yields all dependencies of this feature.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
featureset(self[, new, autoinst, err_missing])	Returns an ordered dict of all features unique in name.

Continued on next page

Table 3 – continued from previous page

fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
gen_dependencies_instances(self[, autoinst, ...])	Checks deps for being instance or class and yields instances.
hide(self[, b])	Hide the feature.
new(self)	Returns new initial feature instance with same parameters.
on_finished(self, source, featureset, sink)	Override this method to be run after extraction.
on_start(self, source, featureset, sink)	Override this method if your feature needs some initialization.
process(self, data, result)	Override this method returning process results.
requires(self)	Override this method if your feature depends on other features.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.
validate_name(self)	Checks for uniqueness of feature name in all dependent features.

get_class_parameters

sigfeat.base.feature.**features_to_featureset**(*features*, *new=False*, *autoinst=False*)

Returns a featureset of given features distinct in names.

Parameters

features [iterable]

new [reinitialize features as new instances.]

autoinst [auto initialize missing feature classes if required.]

3.3 Result

class sigfeat.base.result.**Result**

Result dict. Behaves 'immutable' to the Feature.process method.

Just a simple dict to hold the results from features.

Methods

clear()	
copy()	
fromkeys(iterable[, value])	Create a new dictionary with keys from iterable and values set to value.
get(self, key[, default])	Return the value for key if key is in the dictionary, else default.
items()	
keys()	
pop()	If key is not found, d is returned if given, otherwise KeyError is raised
popitem()	2-tuple; but raise KeyError if D is empty.

Continued on next page

Table 4 – continued from previous page

setdefault(self, key[, default])	Insert key with a value of default if key is not in the dictionary.
update()	If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
values()	

3.4 Sink

Implements the abstract Sink class.

class sigfeat.base.sink.**Sink**
Sink base class.

Methods

receive(datad)	Shall receive dictionaries directly written to source.
receive_append(resultd)	Shall receive result dictionaries appending data to fields.

classmethod receive(*datad*)
Shall receive dictionaries directly written to source.

classmethod receive_append(*resultd*)
Shall receive result dictionaries appending data to fields.

3.5 Preprocess

Implements a simple preprocess base class.

For processing blocks of a source before extracting features. Preprocess behaves like a Source and is consumed by extractor or another Preprocess.

class sigfeat.base.preprocess.**Preprocess**(*source*, ***parameters*)
Preprocess base class.

Behaves like a source. But you mus ovrride the process method.

Examples

```
>>> src = YourPreprocess(YourSource(...))
>>> extractor.extract(src, ...)
```

Attributes

metadata Returns metadata.

parameters Returns all parameters.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.
generate(self)	Override this method.
process(self, data)	Override this method.
unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.

get_class_parameters

generate (*self*)
Override this method. It must yield `data`.

Usually `data = (block, index)`

process (*self*, *data*)
Override this method.

Parameters

data [data from source]

3.6 Parameter

This module implements the Parameter and it's mixin class.

Purpose is, to distinguish parameters of objects from other attributes. Parameters of instances will be extracted into Sink.

class sigfeat.base.parameter.**Parameter** (*default=None*)
Adds a Parameter to the class.

Attributes

default Returns the default value.

Methods

validate(self, value)	You can override this method for validation of parameter values.

default
Returns the default value.

validate (*self*, *value*)
You can override this method for validation of parameter values.

class sigfeat.base.parameter.**ParameterMixin** (***parameters*)
ParameterMixin class

Adds Parameter functionality to classes.

Attributes

parameters	Returns all parameters.

Methods

unroll_parameters(self, parameters)	This method must be called to collect all parameters and provide them as attributes.

get_class_parameters	

parameters
Returns all parameters.

unroll_parameters (*self*, *parameters*)
This method must be called to collect all parameters and provide them as attributes.

By calling this function defined parameters will become attributes with values not beeing of type Parameter anymore. But all parameters will be placed in self._parameters as well.

3.7 Metadata

Mixin for Metadata.

Adds methods `add_metadata()`, `extend_metadata()` and `fetch_metadata_as_attrs()` to the class.

Metadata will be extracted into Sink. So created feature datasets contain metadata for e.g. features and sources.

class sigfeat.base.metadata.**MetadataMixin**
MetadataMixin class Adding metadata functionality to classes. Overrides:

- `._metadata`
- `.metadata`
- `._init_metadata_list`
- `.add_metadata`
- `.extend_metadata`
- `.fetch_metadata_as_attrs`

Attributes

metadata	Returns metadata.

Methods

add_metadata(self, name, value)	Appends the key value pair to metadata list.
extend_metadata(self, mdata)	Extends the metadata with the given list of key value pairs.
fetch_metadata_as_attrs(self)	Sets metadata as attributes of self.

add_metadata (*self*, *name*, *value*)
Appends the key value pair to metadata list.

extend_metadata (*self*, *mdata*)
Extends the metadata with the given list of key value pairs.

fetch_metadata_as_attrs (*self*)
Sets metadata as attributes of self.

metadata
Returns metadata.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

CHAPTER 5

License

Python Module Index

S

Index

P

R

S

U

V

Z