pycast Documentation

Release 0.0.7-prealpha

Christian Schwarz

November 17, 2012

CONTENTS

PYCAST.COMMON

pycast.common.profileme.**profileMe**
  alias of _ProfileDecorator

pycast.common.helper.**linear_interpolation**(*first*, *last*, *steps*)
  Interpolates all missing values using linear interpolation.

  Parameters

   • **first** (*Numeric*) – Starting value for the interpolation.

   • **last** (*Numeric*) – End Value for the interpolation

   • **steps** (*Integer*) – Number of missing values that have to be calculated.

  Returns Returns a list of floats containing only the missing values.

  Return type List

TIMESERIES

class `pycast.common.timeseries.`**`TimeSeries`**(*isNormalized=False*, *isSorted=False*)
   Represents the base class for all time series data.

   > **Warning**   TimeSeries instances are NOT threadsafe.

   `__add__`(*otherTimeSeries*)
      Creates a new TimeSeries instance containing the data of self and otherTimeSeries.

      > **Parameters**   **otherTimeSeries** (*TimeSeries*) – TimeSeries instance that will be merged with self.

      > **Returns**   Returns a new TimeSeries instance containing the data entries of self and otherTime-Series. This TimeSeries will be sorted.

      > **Return type**   TimeSeries

   `__eq__`(*otherTimeSeries*)
      Returns if the TimeSeries equals another one.

      TimeSeries are equal to each other if:

         • they contain the same number of entries

         • that each data entry in one TimeSeries is also member of the other one.

      The sort order within the TimeSeries datapoints does not matter!

      > **Returns**   `True` if the TimeSeries objects are equal, `False` otherwise.

      > **Return type**   Boolean

   `__getitem__`(*index*)
      Returns the item stored at the TimeSeries index-th position.

      > **Parameters**   **index** (*Integer*) – Position of the element that should be returned. Starts at 0

      > **Returns**   Returns a list consisting of [timestamp, data].

      > **Return type**   List

      > **Raise**   Raises an `IndexError` if the index is out of range.

   `__init__`(*isNormalized=False*, *isSorted=False*)
      Initializes the TimeSeries.

      > **Parameters**

         • **isNormalized** (*Boolean*) – Within a normalized TimeSeries, all data points have the same temporal distance to each other. When this is `True`, the memory consumption of the

TimeSeries might be reduced. Also some algorithms will probably run faster on normalized TimeSeries. This should only be set to `True`, if the TimeSeries is realy normalized! TimeSeries normalization can be forced by executing `TimeSeries.normalize()`.

- **isSorted** (*Boolean*) – If all data points added to the time series are added in their ascending temporal order, this should set to `True`.

__iter__()
Returns an iterator to the TimeSeries stored data.

> **Returns** Returns an iterator for the TimeSeries.
>
> **Return type** Iterator

__len__()
Returns the number of data entries that are part of the time series.

> **Returns** Returns an Integer representing the number on data entries stored

within the TimeSeries. :rtype: Integer

__setitem__(*index*, *value*)
Sets the item at the index-th position of the TimeSeries.

> **Parameters**
>
> - **index** (*Integer*) – Index of the element that should be set.
>
> - **value** (*List*) – A list of the form [timestamp, data]
>
> **Raise** Raises an `IndexError` if the index is out of range.

__str__()
Returns a string representation of the TimeSeries.

> **Returns** Returns a string representing the TimeSeries in the format: TimeSeries([timestamp, data], [timestamp, data], [timestamp, data]).
>
> **Return type** String

__weakref__
list of weak references to the object (if defined)

add_entry(*timestamp*, *data*, *format=None*)
Adds a new data entry to the TimeSeries.

> **Parameters**
>
> - **timestamp** – Time stamp of the datas occurence. This has either to be a float representing the UNIX epochs or a string containing a timestamp in the given format.
>
> - **data** – Data points information. This has to be a numeric value for now.
>
> - **format** (*String*) – Format of the given timestamp. This is used to convert the timestamp into UNIX epochs, if necessary. For valid examples take a look into the `time.strptime()` documentation.

apply(*method*)
Applies the given ForecastingAlgorithm or SmoothingMethod from the `pycast.methods` module to the TimeSeries.

> **Parameters method** (*BaseMethod*) – Method that should be used with the TimeSeries. For more information about the methods take a look into their corresponding documentation.

classmethod **convert_epoch_to_timestamp**(*timestamp*, *format*)
Converts the given float representing UNIX-epochs into an actual timestamp.

Parameters

- **timestamp** (*Float*) – Timestamp in the defined format.

- **format** (*String*) – Format of the given timestamp. This is used to convert the timestamp into UNIX epochs, if necessary. For valid examples take a look into the `time.strptime()` documentation.

Returns Returns an timestamp as defined by format.

Return type String

classmethod **convert_timestamp_to_epoch**(*timestamp*, *format*)
Converts the given timestamp into a float representing UNIX-epochs.

Parameters

- **timestamp** (*Float*) – Timestamp in the defined format.

- **format** (*String*) – Format of the given timestamp. This is used to convert the timestamp into UNIX epochs, if necessary. For valid examples take a look into the `time.strptime()` documentation.

Returns Returns an float, representing the UNIX-epochs for the given timestamp.

Return type Float

classmethod **from_json**(*jsonBaseString*, *format=None*)
Creates a new TimeSeries instance from the given json string.

Parameters

- **jsonBaseString** (*String*) – JSON string, containing the time series data. This should be a string created by `TimeSeries.to_json()`.

- **format** (*String*) – Format of the given timestamp. This is used to convert the timestamp into UNIX epochs, if necessary. For valid examples take a look into the `time.strptime()` documentation.

Returns Returns a TimeSeries instance containing the data.

Return type TimeSeries

Warning This is an unsafe version! Only use it with the original version. All assumtions regarding normalization and sort order will be ignored and set to default.

classmethod **from_twodim_list**(*datalist*, *format=None*, *isSorted=False*)
Initializes the TimeSeries's data from the two dimensional list.

Parameters

- **datalist** (*List*) – List containing multiple iterables with at least two values. The first item will always be used as timestamp in the predefined format, the second represents the value. All other items in those sublists will be ignored.

- **format** (*String*) – Format of the given timestamp. This is used to convert the timestamp into UNIX epochs, if necessary. For valid examples take a look into the `time.strptime()` documentation.

- **isSorted** (*Boolean*) – Determines if the datalist is sorted by the timestamps. If this is False, the TimeSeries instance sorts itself after all values are read.

Returns Returns a TimeSeries instance containing the data from datalist.

Return type TimeSeries

initialize_from_sql_cursor(*sqlcursor*, *format=None*, *isSorted=False*)
>   Initializes the TimeSeries's data from the given SQL cursor.

>   **Parameters**

>   - **sqlcursor** (*SQLCursor*) – Cursor that was holds the SQL result for any given "SELECT timestamp, value, ... FROM ..." SQL query. Only the first two attributes of the SQL result will be used.

>   - **format** (*String*) – Format of the given timestamp. This is used to convert the timestamp into UNIX epochs, if necessary. For valid examples take a look into the `time.strptime()` documentation.

>   - **isSorted** (*Boolean*) – Determines if the SQL result is already sorted. If this is False, the TimeSeries instance sorts itself after all values are read.

>   **Returns** Returns the number of entries added to the TimeSeries.

>   **Return type** Integer

>   **Todo** This function is not bulletprove, yet.

is_normalized()
>   Returns if the TimeSeries is normalized.

>   **Returns** Returns `True` if the TimeSeries is normalized, `False` otherwise.

>   **Return type** Boolean

is_sorted()
>   Returns if the TimeSeries is sorted.

>   **Returns** Returns `True` if the TimeSeries is sorted ascending, `False` otherwise.

>   **Return type** Boolean

normalize(*normalizationLevel='minute'*, *fusionMethod='average'*, *interpolationMethod='linear'*)
>   Normalizes the TimeSeries data points.

>   If this function is called, the TimeSeries gets ordered ascending automatically. The new timestamps will represent the center of each time bucket.

>   **Parameters**

>   - **normalizationLevel** (*String*) – Level of normalization that has to be applied. The available normalization levels are defined in `timeseries.NormalizationLevels`.

>   - **fusionMethod** (*String*) – Normalization method that has to be used if multiple data entries exist

>   within the same normalization bucket. The available methods are defined in `timeseries.FusionMethods`. :param String interpolationMethod: Interpolation method that is used if a data entry at a specific time

>   is missing. The available interpolation methods are defined in `timeseries.InterpolationMethods`.

>   **Raise** Raises a `ValueError` if a parameter has an unknown method.

sort_timeseries(*ascending=True*)
>   Sorts the data points within the TimeSeries according to their occurence inline.

Parameters ascending (*Boolean*) – Determines if the TimeSeries will be ordered ascending or decending. If this is set to decending once, the ordered parameter defined in `TimeSeries.__init__()` will be set to False FOREVER.

Returns Returns self for convenience.

Return type TimeSeries

sorted_timeseries(*ascending=True*)
    Returns a sorted copy of the TimeSeries, preserving the original one.

    As an assumtion this new TimeSeries is not ordered anymore by default.

    Parameters ascending (*Boolean*) – Determines if the TimeSeries will be ordered ascending or decending.

    Returns Returns a new TimeSeries instance sorted in the requested order.

    Return type TimeSeries

to_gnuplot_datafile(*datafilepath*, *format=None*)
    Dumps the TimeSeries into a gnuplot compatible data file.

    Parameters

    - **datafilepath** (*String*) – Path used to create the file. If that file already exists, it will be overwritten!

    - **format** (*String*) – Format of the timestamp. This is used to convert the timestamp from UNIX epochs, if necessary. For valid examples take a look into the `time.strptime()` documentation.

    Returns Returns `True` if the data could be written, `False` otherwise.

    Return type Boolean

to_json(*format=None*)
    Returns a JSON representation of the TimeSeries data.

    Parameters format (*String*) – Format of the given timestamp. This is used to convert the timestamp into UNIX epochs, if necessary. For valid examples take a look into the `time.strptime()` documentation.

    Returns Returns a basestring, containing the JSON representation of the current

    data stored within the TimeSeries. :rtype: String

to_twodim_list(*format=None*)
    Serializes the TimeSeries data into a two dimensional list of [timestamp, value] pairs.

    Parameters format (*String*) – Format of the timestamp. This is used to convert the timestamp from UNIX epochs, if necessary. For valid examples take a look into the `time.strptime()` documentation.

    Returns Returns a two dimensional list containing [timestamp, value] pairs.

    Return type List

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

p

INDEX