pyowm Documentation

Release

Claudio Sparpaglione

Jul 11, 2017

Contents

Contents:

CHAPTER 1

pyowm package

Subpackages

pyowm.abstractions package

Submodules

pyowm.abstractions.jsonparser module

Module containing an abstract base class for JSON OWM web API responses parsing

class pyowm.abstractions.jsonparser.**JSONParser**
Bases: object

A global abstract class representing a JSON to object parser.

parse_JSON(*JSON_string*)
Returns a proper object parsed from the input JSON_string. Subclasses know from their specific type which object is to be parsed and returned

Parameters **JSON_string** (*str*) – a JSON text string

Returns an object

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the resulting object

pyowm.abstractions.linkedlist module

Module containing abstractions for defining a linked list data structure

class pyowm.abstractions.linkedlist.**LinkedList**
Bases: object

An abstract class representing a Linked List data structure. Each element in the list should contain data and a reference to the next element in the list.

add(*data*)
Adds a new node to the list. Implementations should decide where to put this new element (at the top, in the middle or at the end of the list) and should therefore update pointers to next elements and the list's size.

Parameters **data** (*object*) – the data to be inserted in the new list node

contains(*data*)
Checks if the provided data is stored in at least one node of the list.

Parameters **data** (*object*) – data of the seeked node

Returns a boolean

index_of(*data*)
Finds the position of a node in the list. The index of the first occurrence of the data is returned (indexes start at 0)

Parameters **data** – data of the seeked node

Type object

Returns the int index or -1 if the node is not in the list

pop()
Removes the last node from the list

Returns the object data that was stored in the last node

remove(*data*)
Removes a node from the list. Implementations should decide the policy to be followed when list items having the same data are to be removed, and should therefore update pointers to next elements and the list's size.

Parameters **data** (*object*) – the data to be removed in the new list node

size()
Returns the number of elements in the list

Returns an int

pyowm.abstractions.owm module

Module containing the abstract PyOWM library main entry point interface

class pyowm.abstractions.owm.**OWM**
Bases: object

A global abstract class representing the OWM web API. Every query to the API is done programmatically via a concrete instance of this class. Subclasses should provide a method for every OWM web API endpoint.

get_API_key()
Returns the OWM API key

Returns the OWM API key string

get_API_version()
Returns the currently supported OWM web API version

Returns the OWM web API version string

get_version()
Returns the current version of the PyOWM library

Returns the current PyOWM library version string

is_API_online()
Returns `True` if the OWM web API is currently online. A short timeout is used to determine API service availability.

Returns bool

set_API_key(*API_key*)
Updates the OWM API key

Parameters **API_key** (*str*) – the new value for the OWM API key

pyowm.abstractions.owmcache module

Module containing the abstract PyOWM cache provider

class pyowm.abstractions.owmcache.**OWMCache**
Bases: object

A global abstract class representing a caching provider which can be used to lookup the JSON responses to the most recently or most frequently issued OWM web API requests. The purpose of the caching mechanism is to avoid OWM web API requests and therefore network traffic: the implementations should be adapted to the time/memory requirements of the OWM data clients (i.e: a "slimmer" cache with lower lookup times but higher miss rates or a "fatter" cache with higher memory consumption and higher hit rates?). Subclasses should implement a proper caching algorithms bearing in mind that different weather data types may have different change rates: in example, observed weather can change very frequently while long-period weather forecasts change less frequently. External caching mechanisms (eg: memcached, redis, etc..) can be used by extending this class into a proper decorator for the correspondent Python bindings.

get(*request_url*)
In case of a hit, returns the JSON string which represents the OWM web API response to the request being identified by a specific string URL.

Parameters **request_url** (*str*) – an URL that uniquely identifies the request whose response is to be looked up

Returns a JSON str in case of cache hit or `None` otherwise

set(*request_url*, *response_json*)
Adds the specified response_json value to the cache using as a lookup key the request_url of the request that generated the value.

Parameters

- **request_url** (*str*) – the request URL
- **response_json** (*str*) – the response JSON

Module contents

pyowm.caches package

Submodules

pyowm.caches.lrucache module

Module containing LRU cache related class

class pyowm.caches.lrucache.**LRUCache**(*cache_max_size=20*, *item_lifetime_millis=600000*)
Bases: *pyowm.abstractions.owmcache.OWMCache*

This cache is made out of a 'table' dict and the 'usage_recency' linked list.'table' maps uses requests' URLs as keys and stores JSON raw responses as values. 'usage_recency' tracks down the "recency" of the OWM web API requests: the more recent a request, the more the element will be far from the "death" point of the recency list. Items in 'usage_recency' are the requests' URLs themselves. The implemented LRU caching mechanism is the following:

- cached elements must expire after a certain time passed into the cache. So when an element is looked up and found in the cache, its insertion timestamp is compared to the current one: if the difference is higher than a prefixed value, then the lookup is considered a MISS: the element is removed either from 'table' and from 'usage_recency' and must be requested again to the OWM web API. If the time difference is ok, then the lookup is considered a HIT.
- when a GET results in a HIT, promote the element to the front of the recency list updating its cache insertion timestamp and return the data to the cache clients
- when a GET results in a MISS, return `None`
- when a SET is issued, check if the maximum size of the cache has been reached: if so, discard the least recently used item from the recency list and the dict; then add the element to 'table' recording its timestamp and finally add it at the front of the recency list.

Parameters

- **cache_max_size** (*int*) – the maximum size of the cache in terms of cached OWM web API responses. A reasonable default value is provided.
- **item_lifetime_millis** (*int*) – the maximum lifetime allowed for a cache item in milliseconds. A reasonable default value is provided.

Returns a new *LRUCache* instance

clean()
Empties the cache

get(*request_url*)
In case of a hit, returns the JSON string which represents the OWM web API response to the request being identified by a specific string URL and updates the recency of this request.

Parameters **request_url** (*str*) – an URL that uniquely identifies the request whose response is to be looked up

Returns a JSON str in case of cache hit or `None` otherwise

set(*request_url*, *response_json*)
Checks if the maximum size of the cache has been reached and in case discards the least recently used item from 'usage_recency' and 'table'; then adds the response_json to be cached to the 'table' dict using

as a lookup key the request_url of the request that generated the value; finally adds it at the front of 'usage_recency'

Parameters

- **request_url** (*str*) – the request URL that uniquely identifies the request whose response is to be cached
- **response_json** (*str*) – the response JSON to be cached

size()
Returns the number of elements that are currently stored into the cache

Returns an int

pyowm.caches.nullcache module

Module containing a null-object cache for OWM web API responses

class pyowm.caches.nullcache.**NullCache**
Bases: *pyowm.abstractions.owmcache.OWMCache*

A null-object implementation of the *OWMCache* abstract class

get(*request_url*)
Always returns None (nothing will ever be cached or looked up!)

Parameters **request_url** (*str*) – the request URL

Returns None

set(*request_url*, *response_json*)
Does nothing.

Parameters

- **request_url** (*str*) – the request URL
- **response_json** (*str*) – the response JSON

Module contents

pyowm.commons package

Submodules

pyowm.commons.frontlinkedlist module

Module containing class related to the implementation of linked-list data structure

class pyowm.commons.frontlinkedlist.**FrontLinkedList**
Bases: *pyowm.abstractions.linkedlist.LinkedList*

Implementation of a linked-list data structure. Insertions are performed at the front of the list and so are O(1) while deletions take O(n) because they can be performed against any of the linked list's elements. Each element in the list is a LinkedListNode instance; after instantiation, the list contains no elements.

Parameters

- **first_node** (LinkedListNode) – reference to the first LinkedListNode element in the list

- **last_node** (LinkedListNode) – reference to the last LinkedListNode element in the list

add(*data*)
Adds a new data node to the front list. The provided data will be encapsulated into a new instance of LinkedListNode class and linked list pointers will be updated, as well as list's size.

Parameters **data** (*object*) – the data to be inserted in the new list node

contains(*data*)
Checks if the provided data is stored in at least one node of the list.

Parameters **data** (*object*) – the seeked data

Returns a boolean

first_node()

index_of(*data*)
Finds the position of a node in the list. The index of the first occurrence of the data is returned (indexes start at 0)

Parameters **data** – data of the seeked node

Type object

Returns the int index or -1 if the node is not in the list

pop()
Removes the last node from the list

remove(*data*)
Removes a data node from the list. If the list contains more than one node having the same data that shall be removed, then the node having the first occurrency of the data is removed.

Parameters **data** (*object*) – the data to be removed in the new list node

size()
Returns the number of elements in the list

Returns an int

class pyowm.commons.frontlinkedlist.**FrontLinkedListIterator**(*obj*)
Bases: object

Iterator over the LinkedListNode elements of a LinkedList class instance. The implementation keeps a copy of the iterated list so avoid concurrency problems when iterating over it. This can nevertheless be memory-consuming when big lists have to be iterated over.

Parameters **obj** (*object*) – the iterable object (LinkedList)

Returns a FrontLinkedListIterator instance

next()
Compatibility for Python 2.x, delegates to function: *__next__()* Returns the next *Weather* item

Returns the next *Weather* item

class pyowm.commons.frontlinkedlist.**LinkedListNode**(*data*, *next_node*)
Class representing an element of the LinkedList

Parameters

- **data** (*object*) – the actual data that this node holds
- **next** (LinkedListNode) – reference to the next LinkedListNode instance in the list

data()
Returns the data in this node

Returns an object

next()
Returns the next LinkedListNode in the list

Returns a LinkedListNode instance

update_next(*linked_list_node*)

Parameters **linked_list_node** (LinkedListNode) – the new reference to the next LinkedListNode element

pyowm.commons.weather_client module

Module containing classes for HTTP client/server interactions

class pyowm.commons.weather_client.**WeatherHttpClient**(*API_key*, *cache*, *subscription_type='free'*)

Bases: object

API_SUBSCRIPTION_SUBDOMAINS = {'pro': 'pro', 'free': 'api'}
An HTTP client class for the OWM web API. The class can leverage a caching mechanism

Parameters

- **API_key** (*Unicode*) – a Unicode object representing the OWM web API key
- **cache** (an *OWMCache* concrete instance) – an *OWMCache* concrete instance that will be used to cache OWM web API responses.
- **subscription_type** (*str*) – the type of OWM web API subscription to be wrapped. The value is used to pick the proper API subdomain for HTTP calls. Defaults to: 'free'

call_API(*API_endpoint_URL*, *params_dict*, *timeout=<object object>*)
Invokes a specific OWM web API endpoint URL, returning raw JSON data.

Parameters

- **API_endpoint_URL** (*str*) – the API endpoint to be invoked
- **params_dict** (*dict*) – a dictionary containing the query parameters to be used in the HTTP request (given as key-value couples in the dict)
- **timeout** (*int*) – how many seconds to wait for connection establishment (defaults to socket._GLOBAL_DEFAULT_TIMEOUT)

Returns a string containing raw JSON data

Raises *APICallError*

pyowm.commons.airpollution_client module

class pyowm.commons.airpollution_client.**AirPollutionHttpClient**(*API_key*, *cache*)
Bases: object

An HTTP client class for the OWM Air Pollution web API. The class can leverage a caching mechanism

Parameters

- **API_key** (*Unicode*) – a Unicode object representing the OWM Air Pollution web API key
- **cache** (an *OWMCache* concrete instance) – an *OWMCache* concrete instance that will be used to cache OWM Air Pollution web API responses.

get_coi (*params_dict*, *timeout=<object object>*)
Invokes the CO Index endpoint

Parameters

- **params_dict** – dict of parameters
- **timeout** (*int*) – how many seconds to wait for connection establishment (defaults to socket._GLOBAL_DEFAULT_TIMEOUT)

Returns a string containing raw JSON data

Raises *ValueError*, *APICallError*

get_no2 (*params_dict*, *timeout=<object object>*)
Invokes the NO2 Index endpoint

Parameters

- **params_dict** – dict of parameters
- **timeout** (*int*) – how many seconds to wait for connection establishment (defaults to socket._GLOBAL_DEFAULT_TIMEOUT)

Returns a string containing raw JSON data

Raises *ValueError*, *APICallError*

get_o3 (*params_dict*, *timeout=<object object>*)
Invokes the O3 Index endpoint

Parameters

- **params_dict** – dict of parameters
- **timeout** (*int*) – how many seconds to wait for connection establishment (defaults to socket._GLOBAL_DEFAULT_TIMEOUT)

Returns a string containing raw JSON data

Raises *ValueError*, *APICallError*

get_so2 (*params_dict*, *timeout=<object object>*)
Invokes the SO2 Index endpoint

Parameters

- **params_dict** – dict of parameters
- **timeout** (*int*) – how many seconds to wait for connection establishment (defaults to socket._GLOBAL_DEFAULT_TIMEOUT)

Returns a string containing raw JSON data

Raises *ValueError*, *APICallError*

pyowm.commons.uv_client module

class pyowm.commons.uv_client.**UltraVioletHttpClient**(*API_key*, *cache*)
Bases: object

An HTTP client class for the OWM UV web API. The class can leverage a caching mechanism

Parameters

- **API_key** (*Unicode*) – a Unicode object representing the OWM UV web API key
- **cache** (an *OWMCache* concrete instance) – an *OWMCache* concrete instance that will be used to cache OWM UV web API responses.

get_uvi(*params_dict*, *timeout=<object object>*)
Invokes the UV Index endpoint

Parameters

- **params_dict** – dict of parameters
- **timeout** (*int*) – how many seconds to wait for connection establishment (defaults to socket._GLOBAL_DEFAULT_TIMEOUT)

Returns a string containing raw JSON data

Raises *ValueError*, *APICallError*

Module contents

pyowm.exceptions package

Submodules

pyowm.exceptions.api_call_error module

Module containing APICallError class

exception pyowm.exceptions.api_call_error.**APICallError**(*message*, *triggering_error=None*)
Bases: *pyowm.exceptions.OWMError*

Error class that represents generic failures when invoking OWM web API, in example due to network errors.

Parameters

- **message** (*str*) – the message of the error
- **triggering_error** (an *Exception* subtype) – optional *Exception* object that triggered this error (defaults to None)

exception pyowm.exceptions.api_call_error.**BadGatewayError**(*message*, *triggering_error=None*)
Bases: *pyowm.exceptions.api_call_error.APICallError*

Error class that represents 502 errors - i.e when upstream backend cannot communicate with API gateways.

Parameters

- **message** (*str*) – the message of the error
- **triggering_error** (an *Exception* subtype) – optional *Exception* object that triggered this error (defaults to None)

pyowm.exceptions.api_response_error module

Module containing APIResponseError class

exception pyowm.exceptions.api_response_error.**APIResponseError**(*message*)
Bases: *pyowm.exceptions.OWMError*

Error class that represents HTTP error status codes in OWM web API responses.

Parameters **cause** (*str*) – the message of the error

Returns a *APIResponseError* instance

pyowm.exceptions.not_found_error module

Module containing NotFoundError class

exception pyowm.exceptions.not_found_error.**NotFoundError**(*message*)
Bases: *pyowm.exceptions.OWMError*

Error class that represents the situation when an entity is not found into a collection of entities.

Parameters **cause** (*str*) – the message of the error

Returns a *NotFoundError* instance

pyowm.exceptions.parse_response_error module

Module containing ParseResponseError class

exception pyowm.exceptions.parse_response_error.**ParseResponseError**(*message*)
Bases: *pyowm.exceptions.OWMError*

Error class that represents failures when parsing payload data in HTTP responses sent by the OWM web API.

Parameters **cause** (*str*) – the message of the error

Returns a *ParseResponseError* instance

pyowm.exceptions.unauthorized_error module

Module containing Unauthorized class

exception pyowm.exceptions.unauthorized_error.**UnauthorizedError**(*message*)
Bases: *pyowm.exceptions.OWMError*

Error class that represents the situation when an entity cannot be retrieved due to user subscription unsufficient capabilities.

Parameters **cause** (*str*) – the message of the error

Returns a *UnauthorizedError* instance

Module contents

Module containing the OWMError class as base for all other OWM errors

exception pyowm.exceptions.**OWMError**
Bases: exceptions.Exception

pyowm.utils package

Submodules

pyowm.utils.temputils module

Module containing utility functions for temperature and wind units conversion

pyowm.utils.temputils.**kelvin_dict_to**(*d*, *target_temperature_unit*)
Converts all the values in a dict from Kelvin temperatures to the specified temperature format.

Parameters

- **d** (*dict*) – the dictionary containing Kelvin temperature values
- **target_temperature_unit** (*str*) – the target temperature unit, may be: 'celsius' or 'fahrenheit'

Returns a dict with the same keys as the input dict and converted temperature values as values

Raises *ValueError* when unknown target temperature units are provided

pyowm.utils.temputils.**kelvin_to_celsius**(*kelvintemp*)
Converts a numeric temperature from Kelvin degrees to Celsius degrees

Parameters **kelvintemp** (*int/long/float*) – the Kelvin temperature

Returns the float Celsius temperature

Raises *TypeError* when bad argument types are provided

pyowm.utils.temputils.**kelvin_to_fahrenheit**(*kelvintemp*)
Converts a numeric temperature from Kelvin degrees to Fahrenheit degrees

Parameters **kelvintemp** (*int/long/float*) – the Kelvin temperature

Returns the float Fahrenheit temperature

Raises *TypeError* when bad argument types are provided

pyowm.utils.temputils.**metric_wind_dict_to_imperial**(*d*)
Converts all the wind values in a dict from meters/sec (metric measurement system) to miles/hour (imperial measurement system) .

Parameters **d** (*dict*) – the dictionary containing metric values

Returns a dict with the same keys as the input dict and values converted to miles/hour

pyowm.utils.timeformatutils module

Module containing utility functions for time formats conversion

class pyowm.utils.timeformatutils.**UTC**
Bases: datetime.tzinfo

dst(*dt*)

tzname(*dt*)

utcoffset(*dt*)

pyowm.utils.timeformatutils.**timeformat**(*timeobject*, *timeformat*)
Formats the specified time object to the target format type.

Parameters

- **timeobject** (int, `datetime.datetime` or ISO8601-formatted string with pattern `YYYY-MM-DD HH:MM:SS+00`) – the object conveying the time value
- **timeformat** (*str*) – the target format for the time conversion. May be: '*unix*' (outputs an int UNIXtime), '*date*' (outputs a `datetime.datetime` object) or '*iso*' (outputs an ISO8601-formatted string with pattern `YYYY-MM-DD HH:MM:SS+00`)

Returns the formatted time

Raises ValueError when unknown timeformat switches are provided or when negative time values are provided

`pyowm.utils.timeformatutils.`**to_ISO8601**(*timeobject*)

Returns the ISO8601-formatted string corresponding to the time value conveyed by the specified object, which can be either a UNIXtime, a `datetime.datetime` object or an ISO8601-formatted string in the format *YYYY-MM-DD HH:MM:SS+00'*.

Parameters **timeobject** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the time value

Returns an ISO8601-formatted string with pattern *YYYY-MM-DD HH:MM:SS+00'*

Raises *TypeError* when bad argument types are provided, *ValueError* when negative UNIXtimes are provided

`pyowm.utils.timeformatutils.`**to_UNIXtime**(*timeobject*)

Returns the UNIXtime corresponding to the time value conveyed by the specified object, which can be either a UNIXtime, a `datetime.datetime` object or an ISO8601-formatted string in the format *YYYY-MM-DD HH:MM:SS+00'*.

Parameters **timeobject** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the time value

Returns an int UNIXtime

Raises *TypeError* when bad argument types are provided, *ValueError* when negative UNIXtimes are provided

`pyowm.utils.timeformatutils.`**to_date**(*timeobject*)

Returns the `datetime.datetime` object corresponding to the time value conveyed by the specified object, which can be either a UNIXtime, a `datetime.datetime` object or an ISO8601-formatted string in the format *YYYY-MM-DD HH:MM:SS+00'*.

Parameters **timeobject** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the time value

Returns a `datetime.datetime` object

Raises *TypeError* when bad argument types are provided, *ValueError* when negative UNIXtimes are provided

pyowm.utils.timeutils module

Module containing utility functions for time values generation/management

`pyowm.utils.timeutils.`**last_hour**(*date=None*)

Gives the `datetime.datetime` object corresponding to the last hour before now or before the specified `datetime.datetime` object.

Parameters **date** (datetime.datetime object) – the date you want an hour to be subtracted from (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**last_month**(*date=None*)

Gives the datetime.datetime object corresponding to the last month before now or before the specified datetime.datetime object. A month corresponds to 30 days.

Parameters **date** (datetime.datetime object) – the date you want a month to be subtracted from (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**last_three_hours**(*date=None*)

Gives the datetime.datetime object corresponding to last three hours before now or before the specified datetime.datetime object.

Parameters **date** (datetime.datetime object) – the date you want three hours to be subtracted from (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**last_week**(*date=None*)

Gives the datetime.datetime object corresponding to the last week before now or before the specified datetime.datetime object. A week corresponds to 7 days.

Parameters **date** (datetime.datetime object) – the date you want a week to be subtracted from (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**last_year**(*date=None*)

Gives the datetime.datetime object corresponding to the last year before now or before the specified datetime.datetime object. A year corresponds to 365 days.

Parameters **date** (datetime.datetime object) – the date you want a year to be subtracted from (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**next_hour**(*date=None*)

Gives the datetime.datetime object corresponding to the next hour from now or from the specified datetime.datetime object.

Parameters **date** (datetime.datetime object) – the date you want an hour to be added (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**next_month**(*date=None*)

Gives the datetime.datetime object corresponding to the next month after now or after the specified datetime.datetime object. A month corresponds to 30 days.

Parameters **date** (datetime.datetime object) – the date you want a month to be added to (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**next_three_hours**(*date=None*)

Gives the datetime.datetime object corresponding to the next three hours from now or from the specified datetime.datetime object.

Parameters **date** (datetime.datetime object) – the date you want three hours to be added (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**next_week**(*date=None*)

Gives the datetime.datetime object corresponding to the next week from now or from the specified datetime.datetime object. A week corresponds to 7 days.

Parameters **date** (datetime.datetime object) – the date you want a week to be added (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**next_year**(*date=None*)

Gives the datetime.datetime object corresponding to the next year after now or after the specified datetime.datetime object. A month corresponds to 30 days.

Parameters **date** (datetime.datetime object) – the date you want a year to be added to (if left None, the current date and time will be used)

Returns a datetime.datetime object

pyowm.utils.timeutils.**now**(*timeformat='date'*)

Returns the current time in the specified timeformat.

Parameters **timeformat** (*str*) – the target format for the time conversion. May be: '*date*' (default - outputs a datetime.datetime object), '*unix*' (outputs a long UNIXtime) or '*iso*' (outputs an ISO8601-formatted string with pattern YYYY-MM-DD HH:MM:SS+00)

Returns the current time value

Raises ValueError when unknown timeformat switches are provided or when negative time values are provided

pyowm.utils.timeutils.**tomorrow**(*hour=None*, *minute=None*)

Gives the datetime.datetime object corresponding to tomorrow. The default value for optional parameters is the current value of hour and minute. I.e: when called without specifying values for parameters, the resulting object will refer to the time = now + 24 hours; when called with only hour specified, the resulting object will refer to tomorrow at the specified hour and at the current minute.

Parameters

- **hour** (*int*) – the hour for tomorrow, in the format *0-23* (defaults to None)
- **minute** (*int*) – the minute for tomorrow, in the format *0-59* (defaults to None)

Returns a datetime.datetime object

Raises *ValueError* when hour or minute have bad values

pyowm.utils.timeutils.**yesterday**(*hour=None*, *minute=None*)

Gives the datetime.datetime object corresponding to yesterday. The default value for optional parameters is the current value of hour and minute. I.e: when called without specifying values for parameters, the resulting object will refer to the time = now - 24 hours; when called with only hour specified, the resulting object will refer to yesterday at the specified hour and at the current minute.

Parameters

- **hour** (*int*) – the hour for yesterday, in the format *0-23* (defaults to None)
- **minute** (*int*) – the minute for yesterday, in the format *0-59* (defaults to None)

Returns a datetime.datetime object

Raises *ValueError* when hour or minute have bad values

pyowm.utils.xmlutils module

Module containing utility functions for generating XML strings

pyowm.utils.xmlutils.**DOM_node_to_XML**(*tree*, *xml_declaration=True*)
Prints a DOM tree to its Unicode representation.

Parameters

- **tree** (an `xml.etree.ElementTree.Element` object) – the input DOM tree
- **xml_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line

Returns Unicode object

pyowm.utils.xmlutils.**annotate_with_XMLNS**(*tree*, *prefix*, *URI*)
Annotates the provided DOM tree with XMLNS attributes and adds XMLNS prefixes to the tags of the tree nodes.

Parameters

- **tree** (an `xml.etree.ElementTree.ElementTree` or `xml.etree.ElementTree.Element` object) – the input DOM tree
- **prefix** (*str*) – XMLNS prefix for tree nodes' tags
- **URI** (*str*) – the URI for the XMLNS definition file

pyowm.utils.xmlutils.**create_DOM_node_from_dict**(*d*, *name*, *parent_node*)
Dumps dict data to an `xml.etree.ElementTree.SubElement` DOM subtree object and attaches it to the specified DOM parent node. The created subtree object is named after the specified name. If the supplied dict is `None` no DOM node is created for it as well as no DOM subnodes are generated for eventual `None` values found inside the dict

Parameters

- **d** (*dict*) – the input dictionary
- **name** (*str*) – the name for the DOM subtree to be created
- **parent_node** (`xml.etree.ElementTree.Element` or derivative objects) – the parent DOM node the newly created subtree must be attached to

Returns `xml.etree.ElementTree.SubElementTree` object

Module contents

pyowm.webapi25 package

Subpackages

pyowm.webapi25.xsd package

Submodules

pyowm.webapi25.xsd.xmlnsconfig module

XMLNS configuration

Module contents

Submodules

pyowm.webapi25.cityidregistry module

class pyowm.webapi25.cityidregistry.**CityIDRegistry**(*filepath_regex*)

MATCHINGS = {'exact': <function <lambda>>, 'like': <function <lambda>>, 'nocase': <function <lambda>>}

id_for(**args*, ***kwargs*)

Returns the long ID corresponding to the first city found that matches the provided city name. The lookup is case insensitive.

Deprecated since version 3.0.0: Use *ids_for()* instead.

Parameters **city_name** (*str*) – the city name whose ID is looked up

Returns a long or None if the lookup fails

ids_for(*city_name*, *country=None*, *matching='nocase'*)

Returns a list of tuples in the form (long, str, str) corresponding to the int IDs and relative toponyms and 2-chars country of the cities matching the provided city name. The rule for identifying matchings is according to the provided *matching* parameter value. If *country* is provided, the search is restricted to the cities of the specified country. :param country: two character str representing the country where to search for the city. Defaults to *None*, which means: search in all countries. :param matching: str among *exact* (literal, case-sensitive matching), *nocase* (literal, case-insensitive matching) and *like* (matches cities whose name contains as a substring the string fed to the function, no matter the case). Defaults to *nocase*. :raises ValueError if the value for *matching* is unknown :return: list of tuples

location_for(**args*, ***kwargs*)

Returns the *Location* object corresponding to the first city found that matches the provided city name. The lookup is case insensitive.

Parameters **city_name** (*str*) – the city name you want a *Location* for

Returns a *Location* instance or None if the lookup fails

Deprecated since version 3.0.0: Use *locations_for()* instead.

locations_for(*city_name*, *country=None*, *matching='nocase'*)

Returns a list of Location objects corresponding to the int IDs and relative toponyms and 2-chars country of the cities matching the provided city name. The rule for identifying matchings is according to the provided *matching* parameter value. If *country* is provided, the search is restricted to the cities of the specified country. :param country: two character str representing the country where to search for the city. Defaults to *None*, which means: search in all countries. :param matching: str among *exact* (literal, case-sensitive matching), *nocase* (literal, case-insensitive matching) and *like* (matches cities whose name contains as a substring the string fed to the function, no matter the case). Defaults to *nocase*. :raises ValueError if the value for *matching* is unknown :return: list of *webapi25.location.Location* objects

pyowm.webapi25.configuration25 module

pyowm.webapi25.forecast module

Module containing weather forecast classes and data structures.

class pyowm.webapi25.forecast.**Forecast**(*interval*, *reception_time*, *location*, *weathers*)
Bases: object

A class encapsulating weather forecast data for a certain location and relative to a specific time interval (forecast for every three hours or for every day)

Parameters

- **interval** (*str*) – the time granularity of the forecast. May be: *'3h'* for three hours forecast or *'daily'* for daily ones
- **reception_time** (*int*) – GMT UNIXtime of the forecast reception from the OWM web API
- **location** (Location) – the *Location* object relative to the forecast
- **weathers** (*list*) – the list of *Weather* objects composing the forecast

Returns a *Forecast* instance

Raises *ValueError* when negative values are provided

count_weathers()
Tells how many *Weather* items compose the forecast

Returns the *Weather* objects total

get(*index*)
Lookups up into the *Weather* items list for the item at the specified index

Parameters **index** (*int*) – the index of the *Weather* object in the list

Returns a *Weather* object

get_interval()
Returns the time granularity of the forecast

Returns str

get_location()
Returns the Location object relative to the forecast

Returns a *Location* object

get_reception_time(*timeformat='unix'*)

Returns the GMT time telling when the forecast was received from the OWM web API

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format YYYY-MM-DD HH:MM:SS+00 '*date* for datetime.datetime object instance

Returns an int or a str

Raises ValueError

get_weathers()
Returns a copy of the *Weather* objects list composing the forecast

Returns a list of *Weather* objects

set_interval(*interval*)
Sets the time granularity of the forecast

Parameters **interval** (*str*) – the time granularity of the forecast, may be "3h" or "daily"

to_JSON()
Dumps object fields into a JSON formatted string

Returns the JSON string

to_XML(*xml_declaration=True*, *xmlns=True*)
Dumps object fields to an XML-formatted string. The 'xml_declaration' switch enables printing of a leading standard XML line containing XML version and encoding. The 'xmlns' switch enables printing of qualified XMLNS prefixes.

Parameters

- **XML_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line
- **xmlns** (*bool*) – if `True` (default) prints full XMLNS prefixes

Returns an XML-formatted string

class pyowm.webapi25.forecast.**ForecastIterator**(*obj*)
Bases: object

Iterator over the list of *Weather* objects encapsulated in a *Forecast* class instance

Parameters **obj** (*object*) – the iterable object

Returns a *ForecastIterator* instance

next()
Compatibility for Python 2.x, delegates to function: *__next__()* Returns the next *Weather* item

Returns the next *Weather* item

pyowm.webapi25.forecaster module

Module containing weather forecast abstraction classes and data structures.

class pyowm.webapi25.forecaster.**Forecaster**(*forecast*)
Bases: object

A class providing convenience methods for manipulating weather forecast data. The class encapsulates a *Forecast* instance and provides abstractions on the top of it in order to let programmers exploit weather forecast data in a human-friendly fashion.

Parameters **forecast** (*Forecast*) – a *Forecast* instance

Returns a *Forecaster* instance

get_forecast()
Returns the *Forecast* instance

Returns the *Forecast* instance

get_weather_at(*timeobject*)
Gives the *Weather* item in the forecast that is closest in time to the time value conveyed by the parameter

Parameters **timeobject** (long/int, `datetime.datetime` or str)) – may be a UNIX time, a `datetime.datetime` object or an ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns a *Weather* object

most_cold()
Returns the *Weather* object in the forecast having the lowest min temperature. The temperature is retrieved using the `get_temperature['temp_min']` call; was 'temp_min' key missing for every *Weather* instance in the forecast, `None` would be returned.

Returns a *Weather* object or `None` if no item in the forecast is eligible

most_hot()
Returns the *Weather* object in the forecast having the highest max temperature. The temperature is retrieved using the `get_temperature['temp_max']` call; was 'temp_max' key missing for every *Weather* instance in the forecast, `None` would be returned.

Returns a *Weather* object or `None` if no item in the forecast is eligible

most_humid()
Returns the *Weather* object in the forecast having the highest humidty.

Returns a *Weather* object or `None` if no item in the forecast is eligible

most_rainy()
Returns the *Weather* object in the forecast having the highest precipitation volume. The rain amount is retrieved via the `get_rain['all']` call; was the 'all' key missing for every *Weather* instance in the forecast,''None'' would be returned.

Returns a *Weather* object or `None` if no item in the forecast is eligible

most_snowy()
Returns the *Weather* object in the forecast having the highest snow volume. The snow amount is retrieved via the `get_snow['all']` call; was the 'all' key missing for every *Weather* instance in the forecast, `None` would be returned.

Returns a *Weather* object or `None` if no item in the forecast is eligible

most_windy()
Returns the *Weather* object in the forecast having the highest wind speed. The snow amount is retrieved via the `get_wind['speed']` call; was the 'speed' key missing for every *Weather* instance in the forecast, `None` would be returned.

Returns a *Weather* object or `None` if no item in the forecast is eligible

when_clouds()
Returns a sublist of the *Weather* list in the forecast, containing only items having clouds as weather condition.

Returns a list of *Weather* objects

when_ends(*timeformat='unix'*)
Returns the GMT time of the end of the forecast coverage, which is the time of the most recent *Weather* item in the forecast

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns a long or a str

Raises *ValueError* when invalid time format values are provided

when_fog()
Returns a sublist of the *Weather* list in the forecast, containing only items having fog as weather condition.

Returns a list of *Weather* objects

when_hurricane()
Returns a sublist of the *Weather* list in the forecast, containing only items having hurricane as weather condition.

Returns a list of *Weather* objects

when_rain()
Returns a sublist of the *Weather* list in the forecast, containing only items having rain as weather condition.

Returns a list of *Weather* objects

when_snow()
Returns a sublist of the *Weather* list in the forecast, containing only items having snow as weather condition.

Returns a list of *Weather* objects

when_starts(*timeformat='unix'*)
Returns the GMT time of the start of the forecast coverage, which is the time of the most ancient *Weather* item in the forecast

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns a long or a str

Raises *ValueError* when invalid time format values are provided

when_storm()
Returns a sublist of the *Weather* list in the forecast, containing only items having storm as weather condition.

Returns a list of *Weather* objects

when_sun()
Returns a sublist of the *Weather* list in the forecast, containing only items having sun as weather condition.

Returns a list of *Weather* objects

when_tornado()
Returns a sublist of the *Weather* list in the forecast, containing only items having tornado as weather condition.

Returns a list of *Weather* objects

will_be_cloudy_at(*timeobject*)
Tells if at the specified time the condition is clouds. The check is performed on the *Weather* item of the forecast which is closest to the time value conveyed by the parameter

Parameters **timeobject** (long/int, `datetime.datetime` or str)) – may be a UNIX time, a `datetime.datetime` object or an ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns boolean

will_be_foggy_at(*timeobject*)
Tells if at the specified time the condition is fog. The check is performed on the *Weather* item of the forecast which is closest to the time value conveyed by the parameter

Parameters **timeobject** (long/int, `datetime.datetime` or str)) – may be a UNIX time, a `datetime.datetime` object or an ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns boolean

will_be_hurricane_at(*timeobject*)
Tells if at the specified time the condition is hurricane. The check is performed on the *Weather* item of the forecast which is closest to the time value conveyed by the parameter

Parameters **timeobject** (long/int, `datetime.datetime` or str)) – may be a UNIX time, a `datetime.datetime` object or an ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns boolean

will_be_rainy_at(*timeobject*)
Tells if at the specified time the condition is rain. The check is performed on the *Weather* item of the forecast which is closest to the time value conveyed by the parameter

Parameters **timeobject** (long/int, `datetime.datetime` or str)) – may be a UNIX time, a `datetime.datetime` object or an ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns boolean

will_be_snowy_at(*timeobject*)
Tells if at the specified time the condition is snow. The check is performed on the *Weather* item of the forecast which is closest to the time value conveyed by the parameter

Parameters **timeobject** (long/int, `datetime.datetime` or str)) – may be a UNIX time, a `datetime.datetime` object or an ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns boolean

will_be_stormy_at(*timeobject*)
Tells if at the specified time the condition is storm. The check is performed on the *Weather* item of the forecast which is closest to the time value conveyed by the parameter

Parameters **timeobject** (long/int, `datetime.datetime` or str)) – may be a UNIX time, a `datetime.datetime` object or an ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns boolean

will_be_sunny_at(*timeobject*)
Tells if at the specified time the condition is sun. The check is performed on the *Weather* item of the forecast which is closest to the time value conveyed by the parameter

Parameters **timeobject** (long/int, `datetime.datetime` or str)) – may be a UNIX time, a `datetime.datetime` object or an ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns boolean

will_be_tornado_at(*timeobject*)
Tells if at the specified time the condition is tornado. The check is performed on the *Weather* item of the forecast which is closest to the time value conveyed by the parameter

Parameters **timeobject** (long/int, `datetime.datetime` or str)) – may be a UNIX time, a `datetime.datetime` object or an ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns boolean

will_have_clouds()
Tells if into the forecast coverage exist one or more *Weather* items related to cloud conditions

Returns boolean

will_have_fog()
Tells if into the forecast coverage exist one or more *Weather* items related to fog conditions

Returns boolean

will_have_hurricane()
Tells if into the forecast coverage exist one or more *Weather* items related to hurricanes

Returns boolean

will_have_rain()
Tells if into the forecast coverage exist one or more *Weather* items related to rain conditions

Returns boolean

will_have_snow()
Tells if into the forecast coverage exist one or more *Weather* items related to snow conditions

Returns boolean

will_have_storm()
Tells if into the forecast coverage exist one or more *Weather* items related to storms

Returns boolean

will_have_sun()
Tells if into the forecast coverage exist one or more *Weather* items related to sun conditions

Returns boolean

will_have_tornado()
Tells if into the forecast coverage exist one or more *Weather* items related to tornadoes

Returns boolean

pyowm.webapi25.forecastparser module

Module containing a concrete implementation for JSONParser abstract class, returning Forecast objects

class pyowm.webapi25.forecastparser.**ForecastParser**
Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building a *Forecast* instance out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)
Parses a *Forecast* instance out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns a *Forecast* instance or `None` if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the JSON string embeds an HTTP status error (this is an OWM web API 2.5 bug)

pyowm.webapi25.historian module

Module containing weather history abstraction classes and data structures.

class pyowm.webapi25.historian.**Historian**(*station_history*)
Bases: object

A class providing convenience methods for manipulating meteostation weather history data. The class encapsulates a *StationHistory* instance and provides abstractions on the top of it in order to let programmers exploit meteostation weather history data in a human-friendly fashion

Parameters **station_history** (*StationHistory*) – a *StationHistory* instance

Returns a *Historian* instance

average_humidity()
Returns the average value in the humidity series

Returns a float

Raises ValueError when the measurement series is empty

average_pressure()
Returns the average value in the pressure series

Returns a float

Raises ValueError when the measurement series is empty

average_rain()
Returns the average value in the rain series

Returns a float

Raises ValueError when the measurement series is empty

average_temperature(*unit='kelvin'*)
Returns the average value in the temperature series

Parameters **unit** (*str*) – the unit of measure for the temperature values. May be among: '*kelvin*' (default), '*celsius*' or '*fahrenheit*'

Returns a float

Raises ValueError when invalid values are provided for the unit of measure or the measurement series is empty

get_station_history()
Returns the *StationHistory* instance

Returns the *StationHistory* instance

humidity_series()
Returns the humidity time series relative to the meteostation, in the form of a list of tuples, each one containing the couple timestamp-value

Returns a list of tuples

max_humidity()
Returns a tuple containing the max value in the humidity series preceeded by its timestamp

Returns a tuple

Raises ValueError when the measurement series is empty

max_pressure()
Returns a tuple containing the max value in the pressure series preceeded by its timestamp

Returns a tuple

Raises ValueError when the measurement series is empty

max_rain()
Returns a tuple containing the max value in the rain series preceeded by its timestamp

Returns a tuple

Raises ValueError when the measurement series is empty

max_temperature(*unit='kelvin'*)
Returns a tuple containing the max value in the temperature series preceeded by its timestamp

Parameters **unit** (*str*) – the unit of measure for the temperature values. May be among: '*kelvin*' (default), '*celsius*' or '*fahrenheit*'

Returns a tuple

Raises ValueError when invalid values are provided for the unit of measure or the measurement series is empty

min_humidity()
Returns a tuple containing the min value in the humidity series preceeded by its timestamp

Returns a tuple

Raises ValueError when the measurement series is empty

min_pressure()
Returns a tuple containing the min value in the pressure series preceeded by its timestamp

Returns a tuple

Raises ValueError when the measurement series is empty

min_rain()
Returns a tuple containing the min value in the rain series preceeded by its timestamp

Returns a tuple

Raises ValueError when the measurement series is empty

min_temperature(*unit='kelvin'*)
Returns a tuple containing the min value in the temperature series preceeded by its timestamp

Parameters **unit** (*str*) – the unit of measure for the temperature values. May be among: '*kelvin*' (default), '*celsius*' or '*fahrenheit*'

Returns a tuple

Raises ValueError when invalid values are provided for the unit of measure or the measurement series is empty

pressure_series()
Returns the atmospheric pressure time series relative to the meteostation, in the form of a list of tuples, each one containing the couple timestamp-value

Returns a list of tuples

rain_series()
Returns the precipitation time series relative to the meteostation, in the form of a list of tuples, each one containing the couple timestamp-value

Returns a list of tuples

temperature_series(*unit='kelvin'*)
Returns the temperature time series relative to the meteostation, in the form of a list of tuples, each one containing the couple timestamp-value

Parameters **unit** (*str*) – the unit of measure for the temperature values. May be among: '*kelvin*' (default), '*celsius*' or '*fahrenheit*'

Returns a list of tuples

Raises ValueError when invalid values are provided for the unit of measure

wind_series()
Returns the wind speed time series relative to the meteostation, in the form of a list of tuples, each one containing the couple timestamp-value

Returns a list of tuples

pyowm.webapi25.location module

Module containing location-related classes and data structures.

class pyowm.webapi25.location.**Location**(*name*, *lon*, *lat*, *ID*, *country=None*)
Bases: object

A class representing a location in the world. A location is defined through a toponym, a couple of geographic coordinates such as longitude and latitude and a numeric identifier assigned by the OWM web API that uniquely spots the location in the world. Optionally, the country specification may be provided.

Further reference about OWM city IDs can be found at: http://bugs.openweathermap.org/projects/api/wiki/Api_2_5_weather#3-By-city-ID

Parameters

- **name** (*Unicode*) – the location's toponym
- **lon** (*int/float*) – the location's longitude, must be between -180.0 and 180.0
- **lat** (*int/float*) – the location's latitude, must be between -90.0 and 90.0
- **ID** (*int*) – the location's OWM city ID
- **country** (*Unicode*) – the location's country (None by default)

Returns a *Location* instance

Raises *ValueError* if lon or lat values are provided out of bounds

get_ID()
Returns the OWM city ID of the location

Returns the int OWM city ID

get_country()
Returns the country of the location

Returns the Unicode country

get_lat()
Returns the latitude of the location

Returns the float latitude

get_lon()
Returns the longitude of the location

Returns the float longitude

get_name()
Returns the toponym of the location

Returns the Unicode toponym

to_JSON()
Dumps object fields into a JSON formatted string

Returns the JSON string

to_XML(*xml_declaration=True*, *xmlns=True*)
Dumps object fields to an XML-formatted string. The 'xml_declaration' switch enables printing of a leading standard XML line containing XML version and encoding. The 'xmlns' switch enables printing of qualified XMLNS prefixes.

Parameters

- **XML_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line
- **xmlns** (*bool*) – if `True` (default) prints full XMLNS prefixes

Returns an XML-formatted string

pyowm.webapi25.location.**location_from_dictionary**(*d*)
Builds a *Location* object out of a data dictionary. Only certain properties of the dictionary are used: if these properties are not found or cannot be read, an error is issued.

Parameters **d** (*dict*) – a data dictionary

Returns a *Location* instance

Raises *KeyError* if it is impossible to find or read the data needed to build the instance

pyowm.webapi25.observation module

Weather observation classes and data structures.

class pyowm.webapi25.observation.**Observation**(*reception_time*, *location*, *weather*)
Bases: object

A class representing the weather which is currently being observed in a certain location in the world. The location is represented by the encapsulated *Location* object while the observed weather data are held by the encapsulated *Weather* object.

Parameters

- **reception_time** (*int*) – GMT UNIXtime telling when the weather obervation has been received from the OWM web API
- **location** (*Location*) – the *Location* relative to this observation
- **weather** (*Weather*) – the *Weather* relative to this observation

Returns an *Observation* instance

Raises *ValueError* when negative values are provided as reception time

get_location()
Returns the *Location* object for this observation

Returns the *Location* object

get_reception_time(*timeformat='unix'*)

Returns the GMT time telling when the observation has been received from the OWM web API

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns an int or a str

Raises ValueError when negative values are provided

get_weather()

Returns the *Weather* object for this observation

Returns the *Weather* object

to_JSON()

Dumps object fields into a JSON formatted string

Returns the JSON string

to_XML(*xml_declaration=True*, *xmlns=True*)

Dumps object fields to an XML-formatted string. The 'xml_declaration' switch enables printing of a leading standard XML line containing XML version and encoding. The 'xmlns' switch enables printing of qualified XMLNS prefixes.

Parameters

- **XML_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line
- **xmlns** (*bool*) – if `True` (default) prints full XMLNS prefixes

Returns an XML-formatted string

pyowm.webapi25.observationlistparser module

Module containing a concrete implementation for JSONParser abstract class, returning lists of Observation objects

class pyowm.webapi25.observationlistparser.**ObservationListParser**

Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building a list of *Observation* instances out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)

Parses a list of *Observation* instances out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns a list of *Observation* instances or `None` if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the OWM API returns a HTTP status error (this is an OWM web API 2.5 bug)

pyowm.webapi25.observationparser module

Module containing a concrete implementation for JSONParser abstract class, returning Observation objects

class pyowm.webapi25.observationparser.**ObservationParser**
Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building an *Observation* instance out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)
Parses an *Observation* instance out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns an *Observation* instance or `None` if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the JSON string embeds an HTTP status error (this is an OWM web API 2.5 bug)

pyowm.webapi25.owm25 module

Module containing the PyOWM library main entry point

class pyowm.webapi25.owm25.**OWM25**(*parsers*, *API_key=None*, *cache=<pyowm.caches.nullcache.NullCache>*, *language='en'*, *subscription_type='free'*)
Bases: *pyowm.abstractions.owm.OWM*

OWM_API_VERSION = '2.5'
OWM subclass providing methods for each OWM web API 2.5 endpoint. The class is instantiated with *jsonparser* subclasses, each one parsing the response payload of a specific API endpoint

Parameters

- **parsers** (*dict*) – the dictionary containing *jsonparser* concrete instances to be used as parsers for OWM web API 2.5 responses
- **API_key** (*str*) – the OWM web API key (defaults to `None`)
- **cache** (an *OWMCache* concrete instance) – a concrete implementation of class *OWMCache* serving as the cache provider (defaults to a *NullCache* instance)
- **language** (*str*) – the language in which you want text results to be returned. It's a two-characters string, eg: "en", "ru", "it". Defaults to: "en"
- **subscription_type** (*str*) – the type of OWM web API subscription to be wrapped. Can be 'free' (free subscription) or 'pro' (paid subscription), Defaults to: 'free'

Returns an *OWM25* instance

city_id_registry()
Gives the *CityIDRegistry* singleton instance that can be used to lookup for city IDs.

Returns a *CityIDRegistry* instance

coindex_around_coords(*lat*, *lon*, *start=None*, *interval=None*)
Queries the OWM web API for Carbon Monoxide values sampled in the surroundings of the provided geocoordinates and in the specified time interval. A *COIndex* object instance is returned, encapsulating a *Location* object and the list of CO samples If *start* is not provided, the latest available CO samples are

retrieved If *start* is provided but *interval* is not, then *interval* defaults to the maximum extent, which is: *year*

Parameters

- **lat** (*int/float*) – the location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – the location's longitude, must be between -180.0 and 180.0
- **start** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the start value of the search time window start (defaults to `None`). If not provided, the latest available CO samples value are retrieved
- **interval** (*str among: 'minute', 'hour', 'day', 'month, 'year'*) – the length of the search time window starting at *start* (defaults to `None`). If not provided, 'year' is used

Returns a *COIndex* instance or `None` if data is not available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* for wrong input values

daily_forecast(*name*, *limit=None*)

Queries the OWM web API for daily weather forecast for the specified location (eg: "London,uk"). A *Forecaster* object is returned, containing a *Forecast* instance covering a global streak of fourteen days by default: this instance encapsulates *Weather* objects, with a time interval of one day one from each other

Parameters

- **name** (*str or unicode*) – the location's toponym
- **limit** (int or `None`) – the maximum number of daily *Weather* items to be retrieved (default is `None`, which stands for any number of items)

Returns a *Forecaster* instance or `None` if forecast data is not available for the specified location

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* if negative values are supplied for limit

daily_forecast_at_coords(*lat*, *lon*, *limit=None*)

Queries the OWM web API for daily weather forecast for the specified geographic coordinate (eg: latitude: 51.5073509, longitude: -0.1277583). A *Forecaster* object is returned, containing a *Forecast* instance covering a global streak of fourteen days by default: this instance encapsulates *Weather* objects, with a time interval of one day one from each other

Parameters

- **lat** (*int/float*) – location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – location's longitude, must be between -180.0 and 180.0
- **limit** (int or `None`) – the maximum number of daily *Weather* items to be retrieved (default is `None`, which stands for any number of items)

Returns a *Forecaster* instance or `None` if forecast data is not available for the specified location

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* if negative values are supplied for limit

daily_forecast_at_id(*id*, *limit=None*)

Queries the OWM web API for daily weather forecast for the specified city ID (eg: 5128581). A *Forecaster*

object is returned, containing a *Forecast* instance covering a global streak of fourteen days by default: this instance encapsulates *Weather* objects, with a time interval of one day one from each other

Parameters

- **id** (*int*) – the location's city ID
- **limit** (int or `None`) – the maximum number of daily *Weather* items to be retrieved (default is `None`, which stands for any number of items)

Returns a *Forecaster* instance or `None` if forecast data is not available for the specified location

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* if negative values are supplied for limit

get_API_key()
Returns the str OWM API key

Returns a str

get_API_version(**args*, ***kwargs*)
Returns the currently supported OWM web API version

Deprecated since version 3.0.0: Will return a tuple instead of a str

Returns str

get_language()
Returns the language in which the OWM web API shall return text results

Returns the language

get_subscription_type()
Returns the OWM API subscription type

Returns the subscription type

get_version(**args*, ***kwargs*)
Returns the current version of the PyOWM library

Deprecated since version 3.0.0: Will return a tuple instead of a str

Returns str

is_API_online()
Returns True if the OWM web API is currently online. A short timeout is used to determine API service availability.

Returns bool

no2index_around_coords(*lat*, *lon*, *start=None*, *interval=None*)
Queries the OWM web API for Nitrogen Dioxide values sampled in the surroundings of the provided geocoordinates and in the specified time interval. A *NO2Index* object instance is returned, encapsulating a *Location* object and the list of NO2 samples If *start* is not provided, the latest available NO2 samples are retrieved If *start* is provided but *interval* is not, then *interval* defaults to the maximum extent, which is: *year*

Parameters

- **lat** (*int/float*) – the location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – the location's longitude, must be between -180.0 and 180.0

- **start** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the start value of the search time window start (defaults to `None`). If not provided, the latest available NO2 samples value are retrieved
- **interval** (*str among: 'minute', 'hour', 'day', 'month, 'year'*) – the length of the search time window starting at *start* (defaults to `None`). If not provided, 'year' is used

Returns a *NO2Index* instance or `None` if data is not available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* for wrong input values

ozone_around_coords(*lat*, *lon*, *start=None*, *interval=None*)

Queries the OWM web API for Ozone (O3) value in Dobson Units sampled in the surroundings of the provided geocoordinates and in the specified time interval. An *Ozone* object instance is returned, encapsulating a *Location* object and the UV intensity value. If *start* is not provided, the latest available ozone value is retrieved. If *start* is provided but *interval* is not, then *interval* defaults to the maximum extent, which is: *year*

Parameters

- **lat** (*int/float*) – the location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – the location's longitude, must be between -180.0 and 180.0
- **start** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the start value of the search time window start (defaults to `None`). If not provided, the latest available Ozone value is retrieved
- **interval** (*str among: 'minute', 'hour', 'day', 'month, 'year'*) – the length of the search time window starting at *start* (defaults to `None`). If not provided, 'year' is used

Returns an *Ozone* instance or `None` if data is not available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* for wrong input values

set_API_key(*API_key*)

Updates the str OWM API key

Parameters **API_key** (*str*) – the new str API key

set_language(*language*)

Sets the language in which the OWM web API shall return text results

Parameters **language** – the new two-characters language (eg: "ru")

so2index_around_coords(*lat*, *lon*, *start=None*, *interval=None*)

Queries the OWM web API for Sulphur Dioxide values sampled in the surroundings of the provided geocoordinates and in the specified time interval. A *SO2Index* object instance is returned, encapsulating a *Location* object and the list of SO2 samples If *start* is not provided, the latest available SO2 samples are retrieved If *start* is provided but *interval* is not, then *interval* defaults to the maximum extent, which is: *year*

Parameters

- **lat** (*int/float*) – the location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – the location's longitude, must be between -180.0 and 180.0

- **start** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the start value of the search time window start (defaults to `None`). If not provided, the latest available SO2 samples value are retrieved
- **interval** (*str among: 'minute', 'hour', 'day', 'month, 'year'*) – the length of the search time window starting at *start* (defaults to `None`). If not provided, 'year' is used

Returns a *SO2Index* instance or `None` if data is not available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* for wrong input values

station_at_coords(*lat*, *lon*, *limit=None*)

Queries the OWM web API for weather stations nearest to the specified geographic coordinates (eg: latitude: 51.5073509, longitude: -0.1277583). A list of *Station* objects is returned, this instance encapsulates a last reported *Weather* object.

Parameters

- **lat** (*int/float*) – location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – location's longitude, must be between -180.0 and 180.0
- **cnt** (int or `None`) – the maximum number of *Station* items to be retrieved (default is `None`, which stands for any number of items)

Returns a list of *Station* objects or `None` if station data is not available for the specified location

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached

station_day_history(*station_ID*, *limit=None*)

Queries the OWM web API for historic weather data measurements for the specified meteostation (eg: 2865), sampled once a day. A *Historian* object instance is returned, encapsulating a *StationHistory* objects which contains the measurements. The total number of retrieved data points can be limited using the appropriate parameter

Parameters

- **station_ID** (*int*) – the numeric ID of the meteostation
- **limit** (int or `None`) – the maximum number of data points the result shall contain (default is `None`, which stands for any number of data points)

Returns a *Historian* instance or `None` if data is not available for the specified meteostation

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* if the limit value is negative

station_hour_history(*station_ID*, *limit=None*)

Queries the OWM web API for historic weather data measurements for the specified meteostation (eg: 2865), sampled once a hour. A *Historian* object instance is returned, encapsulating a *StationHistory* objects which contains the measurements. The total number of retrieved data points can be limited using the appropriate parameter

Parameters

- **station_ID** (*int*) – the numeric ID of the meteostation
- **limit** (int or `None`) – the maximum number of data points the result shall contain (default is `None`, which stands for any number of data points)

Returns a *Historian* instance or `None` if data is not available for the specified meteostation

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* if the limit value is negative

station_tick_history(*station_ID*, *limit=None*)

Queries the OWM web API for historic weather data measurements for the specified meteostation (eg: 2865), sampled once a minute (tick). A *StationHistory* object instance is returned, encapsulating the measurements: the total number of data points can be limited using the appropriate parameter

Parameters

- **station_ID** (*int*) – the numeric ID of the meteostation
- **limit** (int or `None`) – the maximum number of data points the result shall contain (default is `None`, which stands for any number of data points)

Returns a *StationHistory* instance or `None` if data is not available for the specified meteostation

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* if the limit value is negative

three_hours_forecast(*name*)

Queries the OWM web API for three hours weather forecast for the specified location (eg: "London,uk"). A *Forecaster* object is returned, containing a *Forecast* instance covering a global streak of five days: this instance encapsulates *Weather* objects, with a time interval of three hours one from each other

Parameters **name** (*str or unicode*) – the location's toponym

Returns a *Forecaster* instance or `None` if forecast data is not available for the specified location

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached

three_hours_forecast_at_coords(*lat*, *lon*)

Queries the OWM web API for three hours weather forecast for the specified geographic coordinate (eg: latitude: 51.5073509, longitude: -0.1277583). A *Forecaster* object is returned, containing a *Forecast* instance covering a global streak of five days: this instance encapsulates *Weather* objects, with a time interval of three hours one from each other

Parameters

- **lat** (*int/float*) – location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – location's longitude, must be between -180.0 and 180.0

Returns a *Forecaster* instance or `None` if forecast data is not available for the specified location

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached

three_hours_forecast_at_id(*id*)

Queries the OWM web API for three hours weather forecast for the specified city ID (eg: 5128581). A *Forecaster* object is returned, containing a *Forecast* instance covering a global streak of five days: this instance encapsulates *Weather* objects, with a time interval of three hours one from each other

Parameters **id** (*int*) – the location's city ID

Returns a *Forecaster* instance or `None` if forecast data is not available for the specified location

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached

uvindex_around_coords(*lat*, *lon*)
Queries the OWM web API for Ultra Violet value sampled in the surroundings of the provided geocoordinates and in the specified time interval. A *UVIndex* object instance is returned, encapsulating a *Location* object and the UV intensity value. If *start* is not provided, the latest available UVIndex value is retrieved. If *start* is provided but *interval* is not, then *interval* defaults to the maximum extent, which is: *year*

Parameters

- **lat** (*int/float*) – the location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – the location's longitude, must be between -180.0 and 180.0

Returns a *UVIndex* instance or `None` if data is not available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* for wrong input values

weather_around_coords(*lat*, *lon*, *limit=None*)
Queries the OWM web API for the currently observed weather in all the locations in the proximity of the specified coordinates.

Parameters

- **lat** (*int/float*) – location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – location's longitude, must be between -180.0 and 180.0
- **limit** – the maximum number of *Observation* items in the returned list (default is `None`, which stands for any number of items)
- **limit** – int or `None`

Returns a list of *Observation* objects or `None` if no weather data is available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* when coordinates values are out of bounds or negative values are provided for limit

weather_at_coords(*lat*, *lon*)
Queries the OWM web API for the currently observed weather at the specified geographic (eg: 51.503614, -0.107331).

Parameters

- **lat** (*int/float*) – the location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – the location's longitude, must be between -180.0 and 180.0

Returns an *Observation* instance or `None` if no weather data is available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed or *APICallException* when OWM web API can not be reached

weather_at_id(*id*)
Queries the OWM web API for the currently observed weather at the specified city ID (eg: 5128581)

Parameters **id** (*int*) – the location's city ID

Returns an *Observation* instance or `None` if no weather data is available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed or *APICallException* when OWM web API can not be reached

weather_at_ids(*ids_list*)
Queries the OWM web API for the currently observed weathers at the specified city IDs (eg: [5128581,87182])

Parameters **ids_list** (*list of int*) – the list of city IDs

Returns a list of *Observation* instances or an empty list if no weather data is available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed or *APICallException* when OWM web API can not be reached

weather_at_place(*name*)

Queries the OWM web API for the currently observed weather at the specified toponym (eg: "London,uk")

Parameters **name** (*str or unicode*) – the location's toponym

Returns an *Observation* instance or `None` if no weather data is available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed or *APICallException* when OWM web API can not be reached

weather_at_places(*pattern*, *searchtype*, *limit=None*)

Queries the OWM web API for the currently observed weather in all the locations whose name is matching the specified text search parameters. A twofold search can be issued: *'accurate'* (exact matching) and *'like'* (matches names that are similar to the supplied pattern).

Parameters

- **pattern** (*str*) – the string pattern (not a regex) to be searched for the toponym
- **searchtype** – the search mode to be used, must be *'accurate'* for an exact matching or *'like'* for a likelihood matching
- **limit** – the maximum number of *Observation* items in the returned list (default is `None`, which stands for any number of items)
- **limit** – int or `None`

Type searchtype: str

Returns a list of *Observation* objects or `None` if no weather data is available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* when bad value is supplied for the search type or the maximum number of items retrieved

weather_at_station(*station_id*)

Queries the OWM web API for the weather currently observed by a specific meteostation (eg: 29584)

Parameters **station_id** (*int*) – the meteostation ID

Returns an *Observation* instance or `None` if no weather data is available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed or *APICallException* when OWM web API can not be reached

weather_at_stations_in_bbox(*lat_top_left*, *lon_top_left*, *lat_bottom_right*, *lon_bottom_right*, *cluster=False*, *limit=None*)

Queries the OWM web API for the weather currently observed by meteostations inside the bounding box of latitude/longitude coords.

Parameters

- **lat_top_left** (*int/float*) – latitude for top-left of bounding box, must be between -90.0 and 90.0
- **lon_top_left** (*int/float*) – longitude for top-left of bounding box must be between -180.0 and 180.0

- **lat_bottom_right** (*int/float*) – latitude for bottom-right of bounding box, must be between -90.0 and 90.0
- **lon_bottom_right** (*int/float*) – longitude for bottom-right of bounding box, must be between -180.0 and 180.0
- **cluster** (*bool*) – use server clustering of points
- **limit** – the maximum number of *Observation* items in the returned list (default is `None`, which stands for any number of items)
- **limit** – int or `None`

Returns a list of *Observation* objects or `None` if no weather data is available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* when coordinates values are out of bounds or negative values are provided for limit

weather_at_zip_code(*zipcode*, *country*)

Queries the OWM web API for the currently observed weather at the specified zip code and country code (eg: 2037, au).

Parameters

- **zip** (*string*) – the location's zip or postcode
- **country** (*string*) – the location's country code

Returns an *Observation* instance or `None` if no weather data is available

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed or *APICallException* when OWM web API can not be reached

weather_history_at_coords(*lat*, *lon*, *start=None*, *end=None*)

Queries the OWM web API for weather history for the specified at the specified geographic (eg: 51.503614, -0.107331). A list of *Weather* objects is returned. It is possible to query for weather history in a closed time period, whose boundaries can be passed as optional parameters.

Parameters

- **lat** (*int/float*) – the location's latitude, must be between -90.0 and 90.0
- **lon** (*int/float*) – the location's longitude, must be between -180.0 and 180.0
- **start** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the time value for the start query boundary (defaults to `None`)
- **end** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the time value for the end query boundary (defaults to `None`)

Returns a list of *Weather* instances or `None` if history data is not available for the specified location

weather_history_at_id(*id*, *start=None*, *end=None*)

Queries the OWM web API for weather history for the specified city ID. A list of *Weather* objects is returned. It is possible to query for weather history in a closed time period, whose boundaries can be passed as optional parameters.

Parameters

- **id** (*int*) – the city ID
- **start** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the time value for the start query boundary (defaults to `None`)

- **end** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the time value for the end query boundary (defaults to `None`)

Returns a list of *Weather* instances or `None` if history data is not available for the specified location

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* if the time boundaries are not in the correct chronological order, if one of the time boundaries is not `None` and the other is or if one or both of the time boundaries are after the current time

weather_history_at_place(*name*, *start=None*, *end=None*)

Queries the OWM web API for weather history for the specified location (eg: "London,uk"). A list of *Weather* objects is returned. It is possible to query for weather history in a closed time period, whose boundaries can be passed as optional parameters.

Parameters

- **name** (*str or unicode*) – the location's toponym
- **start** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the time value for the start query boundary (defaults to `None`)
- **end** (int, `datetime.datetime` or ISO8601-formatted string) – the object conveying the time value for the end query boundary (defaults to `None`)

Returns a list of *Weather* instances or `None` if history data is not available for the specified location

Raises *ParseResponseException* when OWM web API responses' data cannot be parsed, *APICallException* when OWM web API can not be reached, *ValueError* if the time boundaries are not in the correct chronological order, if one of the time boundaries is not `None` and the other is or if one or both of the time boundaries are after the current time

pyowm.webapi25.station module

Module containing classes and data structures related to meteostation data

class `pyowm.webapi25.station.`**Station**(*name*, *station_ID*, *station_type*, *status*, *lat*, *lon*, *distance=None*, *last_weather=None*)

Bases: `object`

A class representing meteostations which are reporting current weather conditions from geographical coordinates.

Parameters

- **name** (*string*) – meteostation name
- **station_ID** (*int*) – OWM station ID
- **station_type** (*int*) – meteostation type
- **status** (*int*) – station status
- **lat** (*float*) – latitude for station
- **lon** (*float*) – longitude for station
- **distance** (*float*) – distance of station from lat/lon of search criteria
- **last_weather** (*Weather* instance) – last reported weather conditions from station

Returns a *Station* instance

Raises *ValueError* if *lon* or *lat* values are provided out of bounds or *last_weather* is not an instance of *Weather* or *None*

get_distance()
Returns the distance of the station from the geo coordinates used in search

Returns the float distance from geo coordinates

get_last_weather()
Returns the last reported weather conditions reported by the station.

Returns the last *Weather* instance reported by station

get_lat()
Returns the latitude of the location

Returns the float latitude

get_lon()
Returns the longitude of the location

Returns the float longitude

get_name()
Returns the name of the station

Returns the Unicode station name

get_station_ID()
Returns the OWM station ID

Returns the int OWM station ID

get_station_type()
Returns the OWM station type

Returns the int OWM station type

get_status()
Returns the OWM station status

Returns the int OWM station status

to_JSON()
Dumps object fields into a JSON formatted string

Returns the JSON string

to_XML(*xml_declaration=True*, *xmlns=True*)
Dumps object fields to an XML-formatted string. The 'xml_declaration' switch enables printing of a leading standard XML line containing XML version and encoding. The 'xmlns' switch enables printing of qualified XMLNS prefixes.

Parameters

- **XML_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line
- **xmlns** (*bool*) – if `True` (default) prints full XMLNS prefixes

Returns an XML-formatted string

pyowm.webapi25.stationhistory module

Module containing classes and datastructures related to meteostation history data

class pyowm.webapi25.stationhistory.**StationHistory**(*station_ID*, *interval*, *reception_time*, *measurements*)

Bases: object

A class representing historic weather measurements collected by a meteostation. Three types of historic meteostation data can be obtained by the OWM web API: ticks (one data chunk per minute) data, hourly and daily data.

Parameters

- **station_ID** (*int*) – the numeric ID of the meteostation
- **interval** (*str*) – the time granularity of the meteostation data history
- **reception_time** (*int*) – GMT UNIXtime of the data reception from the OWM web API
- **measurements** (*dict*) – a dictionary containing raw weather measurements

Returns a *StationHistory* instance

Raises *ValueError* when the supplied value for reception time is negative

get_interval()

Returns the interval of the meteostation history data

Returns the int interval

get_measurements()

Returns the measurements of the meteostation as a dict. The dictionary keys are UNIX timestamps and for each one the value is a dict containing the keys 'temperature','humidity','pressure','rain','wind' along with their corresponding numeric values. Eg: `{1362933983: { "temperature": 266.25, "humidity": 27.3, "pressure": 1010.02, "rain": None, "wind": 4.7}, ... }`

Returns the dict containing the meteostation's measurements

get_reception_time(*timeformat='unix'*)

Returns the GMT time telling when the meteostation history data was received from the OWM web API

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns an int or a str

Raises ValueError

get_station_ID()

Returns the ID of the meteostation

Returns the int station ID

set_interval(*interval*)

Sets the interval of the meteostation history data

Parameters **interval** (*string*) – the time granularity of the meteostation history data, may be among "tick","hour" and "day"

set_station_ID(*station_ID*)

Sets the numeric ID of the meteostation

Parameters **station_ID** (*int*) – the numeric ID of the meteostation

to_JSON()
Dumps object fields into a JSON formatted string

Returns the JSON string

to_XML(*xml_declaration=True*, *xmlns=True*)
Dumps object fields to an XML-formatted string. The 'xml_declaration' switch enables printing of a leading standard XML line containing XML version and encoding. The 'xmlns' switch enables printing of qualified XMLNS prefixes.

Parameters

- **XML_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line
- **xmlns** (*bool*) – if `True` (default) prints full XMLNS prefixes

Returns an XML-formatted string

pyowm.webapi25.stationhistoryparser module

Module containing a concrete implementation for JSONParser abstract class, returning a StationHistory instance

class pyowm.webapi25.stationhistoryparser.**StationHistoryParser**
Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building a *StationHistory* instance out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)
Parses a *StationHistory* instance out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns a *StationHistory* instance or `None` if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the JSON string embeds an HTTP status error (this is an OWM web API 2.5 bug)

pyowm.webapi25.stationlistparser module

Module containing a concrete implementation for JSONParser abstract class, returning a list of Station instances

class pyowm.webapi25.stationlistparser.**StationListParser**
Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building a list of *Station* instances out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)
Parses a list of *Station* instances out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns a list of *Station* instances or `None` if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the OWM API returns a HTTP status error (this is an OWM web API 2.5 bug)

pyowm.webapi25.stationparser module

Module containing a concrete implementation for JSONParser abstract class, returning a Station instance

class pyowm.webapi25.stationparser.**StationParser**
Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building a *Station* instance out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)
Parses a *Station* instance out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns a *Station* instance or None if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the JSON string embeds an HTTP status error (this is an OWM web API 2.5 bug)

pyowm.webapi25.weather module

Module containing weather data classes and data structures.

class pyowm.webapi25.weather.**Weather**(*reference_time*, *sunset_time*, *sunrise_time*, *clouds*, *rain*, *snow*, *wind*, *humidity*, *pressure*, *temperature*, *status*, *detailed_status*, *weather_code*, *weather_icon_name*, *visibility_distance*, *dewpoint*, *humidex*, *heat_index*)
Bases: object

A class encapsulating raw weather data. A reference about OWM weather codes and icons can be found at: http://bugs.openweathermap.org/projects/api/wiki/Weather_Condition_Codes

Parameters

- **reference_time** (*int*) – GMT UNIX time of weather measurement
- **sunset_time** (*int or None*) – GMT UNIX time of sunset or None on polar days
- **sunrise_time** (*int or None*) – GMT UNIX time of sunrise or None on polar nights
- **clouds** (*int*) – cloud coverage percentage
- **rain** (*dict*) – precipitation info
- **snow** (*dict*) – snow info
- **wind** (*dict*) – wind info
- **humidity** (*int*) – atmospheric humidity percentage
- **pressure** (*dict*) – atmospheric pressure info
- **temperature** (*dict*) – temperature info
- **status** (*Unicode*) – short weather status
- **detailed_status** (*Unicode*) – detailed weather status
- **weather_code** (*int*) – OWM weather condition code
- **weather_icon_name** (*Unicode*) – weather-related icon name

- **visibility_distance** (*float*) – visibility distance
- **dewpoint** (*float*) – dewpoint
- **humidex** (*float*) – Canadian humidex
- **heat_index** (*float*) – heat index

Returns a *Weather* instance

Raises *ValueError* when negative values are provided

get_clouds()
Returns the cloud coverage percentage as an int

Returns the cloud coverage percentage

get_detailed_status()
Returns the detailed weather status as a Unicode string

Returns the detailed weather status

get_dewpoint()
Returns the dew point as a float

Returns the dew point

get_heat_index()
Returns the heat index as a float

Returns the heat index

get_humidex()
Returns the Canadian humidex as a float

Returns the Canadian humidex

get_humidity()
Returns the atmospheric humidity as an int

Returns the humidity

get_pressure()
Returns a dict containing atmospheric pressure info

Returns a dict containing pressure info

get_rain()
Returns a dict containing precipitation info

Returns a dict containing rain info

get_reference_time(*timeformat='unix'*)
Returns the GMT time telling when the weather was measured

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns an int or a str

Raises ValueError when negative values are provided

get_snow()
Returns a dict containing snow info

Returns a dict containing snow info

get_status()
Returns the short weather status as a Unicode string

Returns the short weather status

get_sunrise_time(*timeformat='unix'*)
Returns the GMT time of sunrise

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time or '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns an int or a str or None

Raises ValueError

get_sunset_time(*timeformat='unix'*)
Returns the GMT time of sunset

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time or '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00`

Returns an int or a str or None

Raises ValueError

get_temperature(*unit='kelvin'*)
Returns a dict with temperature info

Parameters **unit** (*str*) – the unit of measure for the temperature values. May be: '*kelvin*' (default), '*celsius*' or '*fahrenheit*'

Returns a dict containing temperature values.

Raises ValueError when unknown temperature units are provided

get_visibility_distance()
Returns the visibility distance as a float

Returns the visibility distance

get_weather_code()
Returns the OWM weather condition code as an int

Returns the OWM weather condition code

get_weather_icon_name()
Returns weather-related icon name as a Unicode string.

Returns the icon name.

get_wind(*unit='meters_sec'*)
Returns a dict containing wind info

Parameters **unit** (*str*) – the unit of measure for the wind values. May be: '*meters_sec*' (default) or '*miles_hour*'

Returns a dict containing wind info

to_JSON()
Dumps object fields into a JSON formatted string

Returns the JSON string

to_XML(*xml_declaration=True*, *xmlns=True*)
Dumps object fields to an XML-formatted string. The 'xml_declaration' switch enables printing of a leading standard XML line containing XML version and encoding. The 'xmlns' switch enables printing of qualified XMLNS prefixes.

Parameters

- **XML_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line
- **xmlns** (*bool*) – if `True` (default) prints full XMLNS prefixes

Returns an XML-formatted string

pyowm.webapi25.weather.**weather_from_dictionary**(*d*)
Builds a *Weather* object out of a data dictionary. Only certain properties of the dictionary are used: if these properties are not found or cannot be read, an error is issued.

Parameters **d** (*dict*) – a data dictionary

Returns a *Weather* instance

Raises *KeyError* if it is impossible to find or read the data needed to build the instance

pyowm.webapi25.weathercoderegistry module

Module containing weather code lookup and resolution classes

class pyowm.webapi25.weathercoderegistry.**WeatherCodeRegistry**(*code_ranges_dict*)
Bases: `object`

A registry class for looking up weather statuses from weather codes.

Parameters **code_ranges_dict** (*dict*) – a dict containing the mapping between weather statuses (eg: "sun","clouds",etc) and weather code ranges

Returns a *WeatherCodeRegistry* instance

status_for(*code*)
Returns the weather status related to the specified weather status code, if any is stored, `None` otherwise.

Parameters **code** (*int*) – the weather status code whose status is to be looked up

Returns the weather status str or `None` if the code is not mapped

pyowm.webapi25.weatherhistoryparser module

Module containing a concrete implementation for JSONParser abstract class, returning a list of Weather objects

class pyowm.webapi25.weatherhistoryparser.**WeatherHistoryParser**
Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building a list of *Weather* instances out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)
Parses a list of *Weather* instances out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns a list of *Weather* instances or `None` if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the JSON string embeds an HTTP status error (this is an OWM web API 2.5 bug)

pyowm.webapi25.weatherutils module

Module containing search and filter utilities for *Weather* objects lists management

pyowm.webapi25.weatherutils.**any_status_is**(*weather_list*, *status*, *weather_code_registry*)
Checks if the weather status code of any of the *Weather* objects in the provided list corresponds to the detailed status indicated. The lookup is performed against the provided *WeatherCodeRegistry* object.

Parameters

- **weathers** (*list*) – a list of *Weather* objects
- **status** (*str*) – a string indicating a detailed weather status
- **weather_code_registry** (*WeatherCodeRegistry*) – a *WeatherCodeRegistry* object

Returns `True` if the check is positive, `False` otherwise

pyowm.webapi25.weatherutils.**filter_by_status**(*weather_list*, *status*, *weather_code_registry*)
Filters out from the provided list of *Weather* objects a sublist of items having a status corresponding to the provided one. The lookup is performed against the provided *WeatherCodeRegistry* object.

Parameters

- **weathers** (*list*) – a list of *Weather* objects
- **status** (*str*) – a string indicating a detailed weather status
- **weather_code_registry** (*WeatherCodeRegistry*) – a *WeatherCodeRegistry* object

Returns `True` if the check is positive, `False` otherwise

pyowm.webapi25.weatherutils.**find_closest_weather**(*weathers_list*, *unixtime*)
Extracts from the provided list of Weather objects the item which is closest in time to the provided UNIXtime.

Parameters

- **weathers_list** (*list*) – a list of *Weather* objects
- **unixtime** (*int*) – a UNIX time

Returns the *Weather* object which is closest in time or `None` if the list is empty

pyowm.webapi25.weatherutils.**is_in_coverage**(*unixtime*, *weathers_list*)
Checks if the supplied UNIX time is contained into the time range (coverage) defined by the most ancient and most recent *Weather* objects in the supplied list

Parameters

- **unixtime** (*int*) – the UNIX time to be searched in the time range
- **weathers_list** (*list*) – the list of *Weather* objects to be scanned for global time coverage

Returns `True` if the UNIX time is contained into the time range, `False` otherwise

pyowm.webapi25.weatherutils.**status_is**(*weather*, *status*, *weather_code_registry*)
Checks if the weather status code of a *Weather* object corresponds to the detailed status indicated. The lookup is performed against the provided *WeatherCodeRegistry* object.

Parameters

- **weather** (*Weather*) – the *Weather* object whose status code is to be checked
- **status** (*str*) – a string indicating a detailed weather status
- **weather_code_registry** (*WeatherCodeRegistry*) – a *WeatherCodeRegistry* object

Returns `True` if the check is positive, `False` otherwise

pyowm.webapi25.uvindex module

class pyowm.webapi25.uvindex.**UVIndex**(*reference_time*, *location*, *value*, *reception_time*)
Bases: object

A class representing the UltraViolet Index observed in a certain location in the world. The location is represented by the encapsulated *Location* object.

Parameters

- **reference_time** (*int*) – GMT UNIXtime telling when the UV data have been measured
- **location** (*Location*) – the *Location* relative to this UV observation
- **value** (*float*) – the observed UV intensity value
- **reception_time** (*int*) – GMT UNIXtime telling when the observation has been received from the OWM web API

Returns an *UVIndex* instance

Raises *ValueError* when negative values are provided as reception time or UV intensity value

get_exposure_risk()
Returns a string stating the risk of harm from unprotected sun exposure for the average adult on this UV observation :return: str

get_location()
Returns the *Location* object for this UV observation

Returns the *Location* object

get_reception_time(*timeformat='unix'*)
Returns the GMT time telling when the UV has been received from the API

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns an int or a str

Raises ValueError when negative values are provided

get_reference_time(*timeformat='unix'*)

Returns the GMT time telling when the UV has been observed from the OWM web API

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns an int or a str

Raises ValueError when negative values are provided

get_value()
Returns the UV intensity for this observation

Returns float

to_JSON()
Dumps object fields into a JSON formatted string

Returns the JSON string

to_XML(*xml_declaration=True*, *xmlns=True*)
Dumps object fields to an XML-formatted string. The 'xml_declaration' switch enables printing of a leading standard XML line containing XML version and encoding. The 'xmlns' switch enables printing of qualified XMLNS prefixes.

Parameters

- **XML_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line
- **xmlns** (*bool*) – if `True` (default) prints full XMLNS prefixes

Returns an XML-formatted string

pyowm.webapi25.uvindex.**uv_intensity_to_exposure_risk**(*uv_intensity*)

pyowm.webapi25.uvindexparser module

Module containing a concrete implementation for JSONParser abstract class, returning UVIndex objects

class pyowm.webapi25.uvindexparser.**UVIndexParser**
Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building an *UVIndex* instance out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)
Parses an *UVIndex* instance out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns an *UVIndex* instance or `None` if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the JSON string embeds an HTTP status error (this is an OWM web API 2.5 bug)

pyowm.webapi25.coindex module

Carbon Monoxide classes and data structures.

class pyowm.webapi25.coindex.**COIndex**(*reference_time*, *location*, *interval*, *co_samples*, *reception_time*)
Bases: `object`

A class representing the Carbon monOxide Index observed in a certain location in the world. The index is made up of several measurements, each one at a different atmospheric pressure. The location is represented by the encapsulated *Location* object.

Parameters

- **reference_time** (*int*) – GMT UNIXtime telling when the CO data has been measured

- **location** (*Location*) – the *Location* relative to this CO observation
- **interval** (*str*) – the time granularity of the CO observation
- **co_samples** (*list of dicts*) – the CO samples
- **reception_time** (*int*) – GMT UNIXtime telling when the CO observation has been received from the OWM web API

Returns an *COIndex* instance

Raises *ValueError* when negative values are provided as reception time, CO samples are not provided in a list

get_co_sample_with_highest_vmr()
Returns the CO sample with the highest Volume Mixing Ratio value :return: dict

get_co_sample_with_lowest_vmr()
Returns the CO sample with the lowest Volume Mixing Ratio value :return: dict

get_co_samples()
Returns the CO samples for this index

Returns list of dicts

get_interval()
Returns the time granularity interval for this CO index measurement

Returns str

get_location()
Returns the *Location* object for this CO index measurement

Returns the *Location* object

get_reception_time(*timeformat='unix'*)
Returns the GMT time telling when the CO observation has been received from the OWM web API

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns an int or a str

Raises ValueError when negative values are provided

get_reference_time(*timeformat='unix'*)
Returns the GMT time telling when the CO samples have been measured

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns an int or a str

Raises ValueError when negative values are provided

is_forecast()
Tells if the current CO observation refers to the future with respect to the current date :return: bool

to_JSON()
Dumps object fields into a JSON formatted string

Returns the JSON string

to_XML(*xml_declaration=True*, *xmlns=True*)
Dumps object fields to an XML-formatted string. The 'xml_declaration' switch enables printing of a leading standard XML line containing XML version and encoding. The 'xmlns' switch enables printing of qualified XMLNS prefixes.

Parameters

- **XML_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line
- **xmlns** (*bool*) – if `True` (default) prints full XMLNS prefixes

Returns an XML-formatted string

pyowm.webapi25.coindexparser module

class pyowm.webapi25.coindexparser.**COIndexParser**
Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building an *COIndex* instance out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)
Parses an *COIndex* instance out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns an *COIndex* instance or `None` if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the JSON string embeds an HTTP status error (this is an OWM web API 2.5 bug)

pyowm.webapi25.ozone module

class pyowm.webapi25.ozone.**Ozone**(*reference_time*, *location*, *interval*, *du_value*, *reception_time*)
Bases: `object`

A class representing the Ozone (O3) data observed in a certain location in the world. The location is represented by the encapsulated *Location* object.

Parameters

- **reference_time** (*int*) – GMT UNIXtime telling when the O3 data have been measured
- **location** (*Location*) – the *Location* relative to this O3 observation
- **du_value** (*float*) – the observed O3 Dobson Units value (reference: http://www.theozonehole.com/dobsonunit.htm)
- **interval** (*str*) – the time granularity of the O3 observation
- **reception_time** (*int*) – GMT UNIXtime telling when the observation has been received from the OWM web API

Returns an *Ozone* instance

Raises *ValueError* when negative values are provided as reception time or du_value

get_du_value()
Returns the O3 Dobson Unit of this observation

Returns float

get_interval()
Returns the time granularity interval for this O3 observation

Returns str

get_location()
Returns the *Location* object for this O3 observation

Returns the *Location* object

get_reception_time(*timeformat='unix'*)
Returns the GMT time telling when the O3 observation has been received from the OWM web API

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns an int or a str

Raises ValueError when negative values are provided

get_reference_time(*timeformat='unix'*)
Returns the GMT time telling when the O3 data have been measured

Parameters **timeformat** (*str*) – the format for the time value. May be: '*unix*' (default) for UNIX time '*iso*' for ISO8601-formatted string in the format `YYYY-MM-DD HH:MM:SS+00` '*date* for `datetime.datetime` object instance

Returns an int or a str

Raises ValueError when negative values are provided

is_forecast()
Tells if the current O3 observation refers to the future with respect to the current date :return: bool

to_JSON()
Dumps object fields into a JSON formatted string

Returns the JSON string

to_XML(*xml_declaration=True*, *xmlns=True*)
Dumps object fields to an XML-formatted string. The 'xml_declaration' switch enables printing of a leading standard XML line containing XML version and encoding. The 'xmlns' switch enables printing of qualified XMLNS prefixes.

Parameters

- **XML_declaration** (*bool*) – if `True` (default) prints a leading XML declaration line
- **xmlns** (*bool*) – if `True` (default) prints full XMLNS prefixes

Returns an XML-formatted string

pyowm.webapi25.ozone_parser module

Module containing a concrete implementation for JSONParser abstract class, returning UVIndex objects

class pyowm.webapi25.ozone_parser.**OzoneParser**
Bases: *pyowm.abstractions.jsonparser.JSONParser*

Concrete *JSONParser* implementation building an *Ozone* instance out of raw JSON data coming from OWM web API responses.

parse_JSON(*JSON_string*)
Parses an *Ozone* instance out of raw JSON data. Only certain properties of the data are used: if these properties are not found or cannot be parsed, an error is issued.

Parameters **JSON_string** (*str*) – a raw JSON string

Returns an *Ozone* instance or `None` if no data is available

Raises *ParseResponseError* if it is impossible to find or parse the data needed to build the result, *APIResponseError* if the JSON string embeds an HTTP status error (this is an OWM web API 2.5 bug)

Module contents

Submodules

pyowm.constants module

Constants for the PyOWM library

Module contents

The PyOWM init file

Author: Claudio Sparpaglione, @csparpa <csparpa@gmail.com>

Platform: platform independent

pyowm.**OWM**(*API_key='b1b15e88fa797225412429c1c50c122a'*, *version='2.5'*, *config_module=None*, *language=None*, *subscription_type=None*)
A parametrized factory method returning a global OWM instance that represents the desired OWM web API version (or the currently supported one if no version number is specified)

Parameters

- **API_key** (*str*) – the OWM web API key (defaults to a test value)
- **version** (*str*) – the OWM web API version. Defaults to `None`, which means use the latest web API version
- **config_module** (*str (eg: 'mypackage.mysubpackage.myconfigmodule')*) – the Python path of the configuration module you want to provide for instantiating the library. Defaults to `None`, which means use the default configuration values for the web API version support you are currently requesting. Please be aware that malformed user-defined configuration modules can lead to unwanted behaviour!
- **language** (*str*) – the language in which you want text results to be returned. It's a two-characters string, eg: "en", "ru", "it". Defaults to: `None`, which means use the default language.
- **subscription_type** (*str*) – the type of OWM web API subscription to be wrapped. Can be 'free' (free subscription) or 'pro' (paid subscription), Defaults to: 'free'

Returns an instance of a proper *OWM* subclass

Raises *ValueError* when unsupported OWM API versions are provided

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

G

H

I

J

K

L

M

N

O

P

R

S

T

U

W

Y