WeatherDB

Release 0.0.2

Max Schmit

Jan 18, 2022

CONTENTS:

CHAPTER

ONE

WEATHERDB - MODULE

author: Max Schmit

The weather-DB module offers an API to interact with the automaticaly filled weather Database.

Depending on the Database user privileges you can use more or less methodes of the classes.

There are 3 different sub modules with their corresponding classes.

- station: Has a class for every type of station. E.g. PrecipitationStation (or StationN). One object represents one Station with one parameter. This object can get used to get the corresponding timeserie. There is also a StationGroup class that groups the three parameters precipitation, temperature and evapotranspiration together for one station. If one parameter is not available this one won't get groupped.
- stations: Is a grouping class for all the stations of one measurement parameter. E.G. PrecipitationStations (or StationsN). Can get used to do actions on all the stations.
- broker: This submodule has only one class Broker. This one is used to do actions on all the stations together. Mainly only used for updating the DB.

1.1 Get started

To get started you need to enter the credentials to access the Database. If this is an account with read only acces, than only those methodes, that read data from the Database are available. Enter those credentials in the secretSettings.py file.

CHAPTER

TWO

WEATHERDB

2.1 weatherDB package

2.1.1 weatherDB.broker module

2.1.2 weatherDB.station module

2.1.3 weatherDB.stations module

2.1.4 Subpackages

weatherDB.lib package

weatherDB.lib.connections module

class weatherDB.lib.connections.**FTP**(*host=''*, *user=''*, *passwd=''*, *acct=''*, *timeout=<object object>*, *source_address=None*, ***, *encoding='utf-8'*)

Bases: ftplib.FTP

login(***kwargs*)

Login, default anonymous.

weatherDB.lib.connections.**check_superuser**(*methode*)

weatherDB.lib.utils module

class weatherDB.lib.utils.**TimestampPeriod**(*start*, *end*)

Bases: object

A class to save a Timespan with a minimal and maximal Timestamp.

COMPARE = {'inner': {0: <built-in function max>, 1: <built-in function min>}, 'outer': {0: <built-in function min>, 1: <built-in function max>}}

__init__(*start*, *end*)

Initiate a TimestampPeriod.

Parameters

- **start** (*pd.Timestamp or similar*) – The start of the Period.
- **end** (*pd.Timestamp or similar*) – The end of the Period.

contains(*other*)

get_period()

get_sql_format_dict(*format='%Y%m%d %H:%M'*)

has_NaT()

has_only_NaT()

inside(*other*)

is_empty()

strftime(*format='%Y-%m-%d %H:%M:%S'*)

union(*other*, *how='inner'*)
Unite 2 TimestampPeriods to one.

Compares the Periods and computes a new one.

Parameters

- **other** (TimestampPeriod) – The other TimestampPeriod with whome to compare.
- **how** (*str, optional*) – How to compare the 2 TimestampPeriods. Can be "inner" or "outer". "inner": the maximal Timespan for both is computed. "outer": The minimal Timespan for both is computed. The default is "inner".

Returns A new TimespanPeriod object uniting both TimestampPeriods.

Return type *TimestampPeriod*

weatherDB.lib.utils.**get_ftp_file_list**(*ftp_conn*, *ftp_folders*)
Get a list of files in the folders with their modification dates.

Parameters

- **ftp_conn** (*ftplib.FTP*) – Ftp connection.
- **ftp_folders** (*list of str or pathlike object*) – The directories on the ftp server to look for files.

Returns A list of Tuples. Every tuple stands for one file. The tuple consists of (filepath, modification date).

Return type list of tuples of strs

Subpackages

weatherDB.lib.max_fun package

weatherDB.lib.max_fun.import_DWD module

A collection of functions to import data from the DWD-CDC Server.

class weatherDB.lib.max_fun.import_DWD.**FTP**(*host=''*, *user=''*, *passwd=''*, *acct=''*, *timeout=<object object>*, *source_address=None*, ***, *encoding='utf-8'*)
Bases: ftplib.FTP

login(***kwargs*)
Login, default anonymous.

weatherDB.lib.max_fun.import_DWD.**dwd_id_to_str**(*id*)
Convert a station id to normal DWD format as str.

Parameters **id** (*int or str*) – The id of the station.

Returns string of normal DWD Station id.

Return type str

weatherDB.lib.max_fun.import_DWD.**get_dwd_data**(*station_id*, *ftp_folder*)
Get the weather data for one station from the DWD server.

Parameters

- **station_id** (*str or int*) – Number of the station to get the weather data from.
- **ftp_folder** (*str*) – the base folder where to look for the stations_id file. e.g. ftp_folder = "climate_environment/CDC/observations_germany/climate/hourly/precipitation/historical/". If the parent folder, where "recent"/"historical" folder is inside, both the historical and recent data gets merged.

Returns The DataFrame of the selected file in the zip folder.

Return type pandas.DataFrame

weatherDB.lib.max_fun.import_DWD.**get_dwd_file**(*zip_filepath*)
Get a DataFrame from one single (zip-)file from the DWD FTP server.

Parameters **zip_filepath** (*str*) – Path to the file on the server. e.g.

- "/climate_environment/CDC/observations_germany/climate/10_minutes/air_temperature/recent/10minutenwerte_T
- "/climate_environment/CDC/derived_germany/soil/daily/historical/derived_germany_soil_daily_historical_73.txt.g

Returns The DataFrame of the selected file in the zip folder.

Return type pandas.DataFrame

weatherDB.lib.max_fun.import_DWD.**get_dwd_meta**(*ftp_folder*, *min_years=0*, *max_hole_d=9999*)
Get the meta file from the ftp_folder on the DWD server.

Downloads the meta file of a given folder. Corrects the meta file of missing files. So if no file for the station is in the folder the meta entry gets deleted. Reset "von_datum" in meta file if there is a biger gap than max_hole_d. Delets entries with less years than min_years.

Parameters

- **ftp_folder** (*str*) – The path to the directory where to search for the meta file. e.g. "climate_environment/CDC/observations_germany/climate/hourly/precipitation/recent/".
- **min_years** (*int, optional*) – filter the list of stations by a minimum amount of years, that they have data for. 0 if the data should not get filtered. Only works if the meta file has a timerange defined, e.g. in "observations". The default is 0.
- **max_hole_d** (*int*) – The maximum amount of days missing in the data allowed. If there are several files for one station and the time hole is biger than this value, the older "von_datum" is overwriten in the meta GeoDataFrame. The default is 2.

Returns a GeoDataFrame of the meta file

Return type geopandas.GeoDataFrame

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W