Meteoserver

Aug 01, 2021

Contents:

1 meteoserver package **1**
- 1.1 Submodules . . . 1
  - 1.1.1 meteoserver.help module . . . 1
  - 1.1.2 meteoserver.sundata module . . . 1
  - 1.1.3 meteoserver.weatherforecast module . . . 3
- 1.2 Module contents . . . 6

2 Indices and tables **7**

Python Module Index **9**

Index **11**

CHAPTER 1

meteoserver package

1.1 Submodules

1.1.1 meteoserver.help module

Functions to print some basic documentation.

meteoserver.help.**print_help_sunData**()
Print a brief description for each of the columns for the dataframes containing solar data.

meteoserver.help.**print_help_weatherforecast**()
Print a brief description for each of the columns for the dataframes containing weather-forecast data.

1.1.2 meteoserver.sundata module

Functions to obtain, read and write four-day sun-forecast ("Zon Actueel") data from Meteoserver.nl.

meteoserver.sundata.**extract_Sun_dataframes_from_dict**(*dataDict*, *numeric*)
Extract the location name, current-data and forecast Pandas dataframes from a data dictionary.

Parameters

- **dataDict** (*dict*) – The name of the data dictionary to convert.
- **numeric** (*bool*) – Convert dataframe content from strings to numeric/datetime format (default=True). Set this to False if you intend to write a JSON file that is (nearly) identical to the original format.

Returns

Tuple containing (location, current, forecast):

- current (df): Pandas dataframe containing current-weather data from a nearby station.

- forecast (df): Pandas dataframe containing forecast data for the specified location (or region?).
- forecast (df): Pandas dataframe containing forecast data for the specified location (or region?).

Return type tuple (str, df, df)

meteoserver.sundata.**read_json_file_sunData**(*fileJSON*, *loc=False*, *numeric=True*)

Read a Meteoserver Sun-data JSON file from disc and return the current-data and forecast dataframes, and optionally the location name.

This uses the "Zon Actueel" Meteoserver data.

Parameters

- **fileJSNO** (*string*) – The name of the JSON file to read.
- **loc** (*bool*) – Return the location name as a third return value (default=False).
- **numeric** (*bool*) – Convert dataframe content from strings to numeric/datetime format (default=True). Set this to False if you intend to write a JSON file that is (nearly) identical to the original format.

Returns

Tuple containing (current, forecast (, location)):

- current (df): Pandas dataframe containing current-weather data from a nearby station.
- forecast (df): Pandas dataframe containing forecast data for the specified location (or region?).
- location (str): The location the data are for.

Return type tuple (df, df (,str))

meteoserver.sundata.**read_json_url_sunData**(*key*, *location*, *loc=False*, *numeric=True*)

Get the Sun data from the Meteoserver server and return the current-data and forecast dataframes and optionally the location name.

This uses the "Zon Actueel" Meteoserver API/data.

Parameters

- **key** (*string*) – The Meteoserver API key.
- **location** (*string*) – The name of the location (in the Netherlands) to obtain data for (e.g. 'De Bilt').
- **loc** (*bool*) – Return the location name as a third return value (default=False).
- **numeric** (*bool*) – Convert dataframe content from strings to numeric/datetime format (default=True). Set this to False if you intend to write a JSON file that is (nearly) identical to the original format.

Returns

Tuple containing (current, forecast (, location)):

- current (df): Pandas dataframe containing current-weather data from a nearby station.
- forecast (df): Pandas dataframe containing forecast data for the specified location (or region?).
- retLoc (str): The name of the location the data are for (only returned if loc=True).

Return type tuple (df, df (,str))

meteoserver.sundata.**write_json_file_sunData**(*fileName*, *location*, *current*, *forecast*)

Write a Meteoserver sun-forecast-data JSON file to disc.

The resulting file has the same format as a downloaded file (barring some spacing).

Parameters

- **fileName** (*string*) – The name of the JSON file to write.
- **location** (*string*) – The location the data are for.
- **current** (*df*) – Pandas dataframe containing current/recent measurements for the specified location (or region).
- **forecast** (*df*) – Pandas dataframe containing sun forecast data for the specified location (or region).

1.1.3 meteoserver.weatherforecast module

Functions to obtain, read and write 2 (HARMONIE) or 4-10 (GFS) day hourly weather-forecast ("Uurverwachting") data from Meteoserver.nl.

meteoserver.weatherforecast.**extract_hourly_forecast_dataframes_from_dict**(*dataDict*, *numeric*)

Extract the location and forecast-data Pandas dataframe from a data dictionary.

Parameters

- **dataDict** (*dict*) – The data dictionary to convert.
- **numeric** (*bool*) – Convert dataframe content from strings to numeric/datetime format (default=True). Set this to False if you intend to write a JSON file that is (nearly) identical to the original format.

Returns

Tuple containing (location, data):

- location (str): Location the data are for.
- data (df): Pandas dataframe containing forecast data for the specified location (or region).

Return type tuple (str, df)

meteoserver.weatherforecast.**read_json_file_weatherforecast**(*fileJSON*, *full=False*, *loc=False*, *numeric=True*)

Read a Meteoserver weather-forecast-data JSON file from disc and return the data as a dataframe.

This uses the "Uurverwachting" Meteoserver data.

Parameters

- **fileJSNO** (*string*) – The name of the JSON file to read.
- **full** (*bool*) – Return the full dataframe (currently 31 columns). If false, obsolescent and duplicate (in non-SI units) columns are removed (currently, 22 columns are returned). Default: False.
- **loc** (*bool*) – Return the location name as a second return value (default=False).
- **numeric** (*bool*) – Convert dataframe content from strings to numeric/datetime format (default=True). Set this to False if you intend to write a JSON file that is (nearly) identical to the original format.

Returns

Tuple containing (data, location):

- data (df): Pandas dataframe containing forecast data for the specified location (or region).
- location (str): The name of the location the data are for (only returned if loc=True) - in this case, the two return values are returned as a tuple).

Return type tuple (df, str)

meteoserver.weatherforecast.**read_json_url_weatherforecast**(*key*, *location*, *model='GFS'*, *full=False*, *loc=False*, *numeric=True*)

Get hourly weather-forecast data from the Meteoserver server and return them as a dataframe.

This uses the "Uurverwachting" Meteoserver API/data.

Parameters

- **key** (*string*) – The Meteoserver API key.
- **location** (*string*) – The name of the location (in the Netherlands) to obtain data for (e.g. 'De Bilt').
- **model** (*string*) – Weather model to use: 'HARMONIE' or 'GFS' (default: GFS)
  - HARMONIE: use high-resolution HARMONIE model for BeNeLux and HiRLAM for the rest of Europe. Hourly predictions up to 48 hours in advance. New data available at 5:30, 11:30, 17:30 and 23:30 CE(S)T.

– GFS: use GFS model for BeNeLux. Hourly predictions for 4 days, then three-hourly predictions for the next 10 days. New data are available at 0:30, 7:30, 12:30 and 18:30 CE(S)T.

- **full** (*bool*) – Return the full dataframe (currently 31 columns). If false, obsolescent and duplicate (in non-SI units) columns are removed (currently, 22 columns are returned). Default: False.
- **loc** (*bool*) – Return the location name as a second return value (default=False).
- **numeric** (*bool*) – Convert dataframe content from strings to numeric/datetime format (default=True). Set this to False if you intend to write a JSON file that is (nearly) identical to the original format.

Returns

Tuple containing (data, retLoc):

- data (df): Pandas dataframe containing forecast data for the specified location (or region).
- retLoc (str): The name of the location the data are for (only returned if loc=True - in this case, the two return values are returned as a tuple).

Return type tuple (df, str)

meteoserver.weatherforecast.**remove_unused_hourly_forecast_columns**(*dataFrame*)

Remove the (probably) unused columns from a weather-forecast dataframe.

This removes the following columns (if they exist):

- obsolescent 'loc' column.
- wind speed/force 'windb' (Beaufort), 'windknp' (knots) and 'windkmh' (km/h) columns, which can be computed from SI 'winds' (m/s) column.
- wind gust columns: 'gustb' (Beaufort), 'gustkt' (knots) and 'gustkmh', which can be computed from SI 'gust' column (m/s).
- air-pressure columns: 'luchtdmmhg' and 'luchtdinhg', which can be computed from SI luchtd (hPa/mbar).

The number of columns is reduced from 27 to 21 for HARMONIE data, and from 31 to 22 for GFS data.

Parameters **dataFrame** (*df*) – Original Pandas dataframe.

Returns Pruned Pandas dataframe.

Return type dataFrame (df)

meteoserver.weatherforecast.**write_json_file_weatherforecast**(*fileName*, *location*, *data*)

Write a Meteoserver weather-forecast-data JSON file to disc.

The resulting file has the same format as a downloaded file (barring some spacing).

Parameters

- **fileName** (*string*) – The name of the JSON file to write.

- **location** (*string*) – The location the data are for.
- **data** (*df*) – Pandas dataframe containing forecast data for the specified location (or region).

1.2 Module contents

Meteoserver module

Meteoserver contains a Python module to obtain and read Dutch weather data from Meteoserver.nl. The code is being developed by Marc van der Sluys of the department of Astrophysics at the Radboud University Nijmegen, the Netherlands and the department of Sustainable energy of the HAN University of Applied Sciences in Arnhem, the Netherlands. The Meteoserver package can be used under the conditions of the GPLv3 licence. These pages contain the API documentation. For more information on the Python package, licence and source code, see the Meteoserver GitHub page.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

meteoserver, 6
meteoserver.help, 1
meteoserver.sundata, 1
meteoserver.weatherforecast, 3

Index

E

extract_hourly_forecast_dataframes_from_dict() (*in module meteoserver.weatherforecast*), 3
extract_Sun_dataframes_from_dict() (*in module meteoserver.sundata*), 1

M

meteoserver (*module*), 6
meteoserver.help (*module*), 1
meteoserver.sundata (*module*), 1
meteoserver.weatherforecast (*module*), 3

P

print_help_sunData() (*in module meteoserver.help*), 1
print_help_weatherforecast() (*in module meteoserver.help*), 1

R

read_json_file_sunData() (*in module meteoserver.sundata*), 2
read_json_file_weatherforecast() (*in module meteoserver.weatherforecast*), 3
read_json_url_sunData() (*in module meteoserver.sundata*), 2
read_json_url_weatherforecast() (*in module meteoserver.weatherforecast*), 4
remove_unused_hourly_forecast_columns() (*in module meteoserver.weatherforecast*), 5

W

write_json_file_sunData() (*in module meteoserver.sundata*), 3
write_json_file_weatherforecast() (*in module meteoserver.weatherforecast*), 5