datedown Documentation

Release 0.2

Christoph Paulik

July 26, 2016

Contents

Small library to download files with date and time based filenames or folder structures. In parallel using wget.

Recursive wget can be slow and result in cumbersome local folder structures. This library downloads exact filenames based on exact dates or a range of dates. Remote and local filenames and paths are built using the Python strftime and strptime format specification

The library uses the Python multiprocessing module to start multiple wget instances for possibly faster downloading. At the end of the download process it verfies that all the files were downloaded. No support for checksums at the moment.

CHAPTER 1

Installation

- Install wget if it is not already on your system.
- `pip install datedown`

CHAPTER 2

Usage

The program can be used either as a library to be called from other Python programs or as a stand alone command line program.

2.1 Use as a command line program

After installation the `datedown` program should be available in your shell. To get detailed instructions on how to use it run `datedown -h`.

If it is impossible to know the exact filename on the server then also a recursive version of the script is available under the name `datedown_rec`.

2.1.1 Example

```
datedown 2000-01-01 2000-01-02 http://localhost:8888 file_%Y_%m_%d.txt /home/cpa/ --urlsubdirs test_
```

This would download the files

- http://localhost:8888/test_data/year_month_subfolders/2000/01/file_2000_01_01.txt
- http://localhost:8888/test_data/year_month_subfolders/2000/01/file_2000_01_02.txt

to

- /home/cpa/test_data/year_month_subfolders/2000/01/file_2000_01_01.txt
- /home/cpa/test_data/year_month_subfolders/2000/01/file_2000_01_02.txt

2.2 Use as a library

For use as a library the most important function is *datedown.interface.download_by_dt()* or *datedown.down.download()*. The first function takes functions that produce urls from Python datetime objects whereas the second takes lists of urls and local filenames. Please see the API Documentation for more details about these functions.

CHAPTER 3

Note

This project has been set up using PyScaffold 2.5.6. For details and usage information on PyScaffold see http://pyscaffold.readthedocs.org/.

CHAPTER 4

Contents

4.1 License

```
The MIT License (MIT)

Copyright (c) 2016 Christoph Paulik

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

4.2 Developers

- Christoph Paulik <christoph.paulik@geo.tuwien.ac.at>

4.3 Changelog

4.3.1 Version 0.2

- Add option for recursive downloads.

4.3.2 Version 0.1

- Initial version.

4.4 datedown

4.4.1 datedown package

Submodules

datedown.dates module

Module for getting date lists in different intervals. This only covers the basics like n-hourly, n-daily and dekadal. For the generation of more complex datetime lists a package like pandas can be used.

`datedown.dates.daily(start, end)`
Iterate over list of daily datetime objects.

Parameters

- **start** (*datetime.datetime*) – first date yielded
- **end** (*datetime.datetime*) – last date yielded

Yields **dt** (*datetime.datetime*) – datetime object between start and end in daily steps.

`datedown.dates.hourly(start, end)`
Iterate over list of hourly datetime objects.

Parameters

- **start** (*datetime.datetime*) – first date yielded
- **end** (*datetime.datetime*) – last date yielded

Yields **dt** (*datetime.datetime*) – datetime object between start and end in daily steps.

`datedown.dates.n_daily(start, end, n)`
Iterate over list of n-daily datetime objects.

Parameters

- **start** (*datetime.datetime*) – first date yielded
- **end** (*datetime.datetime*) – last date yielded
- **n** (*int*) – step size

Yields **dt** (*datetime.datetime*) – datetime object between start and end in daily steps.

`datedown.dates.n_hourly(start, end, n)`
Iterate over list of n-hourly datetime objects.

Parameters

- **start** (*datetime.datetime*) – first date yielded
- **end** (*datetime.datetime*) – last date yielded
- **n** (*int*) – number of hours between each yielded datetime object.

Yields **dt** (*datetime.datetime*) – datetime object between start and end in n-hourly steps.

datedown.down module

Module that puts the things toghether.

datedown.down.**check_downloaded**(*urls*, *targets*)
Check if files that should be downloaded exist. If not then return a list of not downloaded URLs.

Parameters

- **urls** (*iterable*) – iterable over url strings
- **targets** (*iterable*) – paths where to store the files

Returns

- **not_urls** (*list*) – list of urls that do not exist locally
- **not_fnames** (*list*) – list of filenames that do not exist locally

datedown.down.**download**(*urls*, *targets*, *num_proc=1*, *username=None*, *password=None*, *recursive=False*)
Download the urls and store them at the target filenames.

Parameters

- **urls** (*iterable*) – iterable over url strings
- **targets** (*iterable*) – paths where to store the files
- **num_proc** (*int, optional*) – Number of parallel downloads to start
- **username** (*string, optional*) – Username to use for login
- **password** (*string, optional*) – Password to use for login
- **recursive** (*boolean, optional*) – If set then no exact filenames can be given. The data will then be downloaded recursively and stored in the target folder.

datedown.fname_creator module

Module for creating the filenames from the datetimes.

datedown.fname_creator.**create_dt_fpath**(*dt*, *root*, *fname*, *subdirs=[]*)
Create filepaths from root + fname and a list of subdirectories. fname and subdirs will be put through dt.strftime.

Parameters

- **dt** (*datetime.datetime*) – date as basis for the URL
- **root** (*string*) – root of the filenpath
- **fname** (*string*) – filename to use
- **subdirs** (*list, optional*) – list of strings. Each element represents a subdirectory. For example the list ['%Y', '%m'] would lead to a URL of `root/YYYY/MM/fname` or for a dt of datetime(2000,12,31) `root/2000/12/fname`

Returns **fpath** – Full filename including path

Return type string

datedown.interface module

Interface for the package.

datedown.interface.**download_by_dt**(*dts*, *url_create_fn*, *fpath_create_fn*, *download_fn*, *passes=3*, *recursive=False*)

Download data for datetimes. If files are missing try again passes times.

Parameters

- **dts** (*list*) – list of datetime.datetime objects
- **url_create_fn** (*function*) – function that creates an URL from a datetime object
- **fpath_create_fn** (*function*) – function that creates a filename from a datetime object
- **download_fn** (*function*) – function that transfers data from a list of URLs to a list of filenames. Takes two arguments (url_list, fname_list)
- **passes** (*int, optional*) – if files are missing then try again passes times
- **recursive** (*boolean, optional*) – If set then no exact filenames can be given. The data will then be downloaded recursively and stored in the target folder. No checking of downloaded files is possible in this case.

datedown.interface.**main**(*args*)

datedown.interface.**main_recursive**(*args*)

datedown.interface.**mkdate**(*datestring*)

datedown.interface.**n_hours**(*intervalstring*)

Convert an interval string like 1D, 6H etc. to the number of hours it represents.

datedown.interface.**parse_args**(*args*)

Parse command line parameters

Parameters **args** – command line parameters as list of strings

Returns command line parameters as argparse.Namespace

datedown.interface.**parse_args_recursive**(*args*)

Parse command line parameters for recursive download

Parameters **args** – command line parameters as list of strings

Returns command line parameters as argparse.Namespace

datedown.interface.**run**()

datedown.interface.**run_recursive**()

datedown.urlcreator module

Module for creating the URLs from the datetimes.

datedown.urlcreator.**create_dt_url**(*dt*, *root*, *fname*, *subdirs=[]*)

Create URLs from root + fname and a list of subdirectories. fname and subdirs will be put through dt.strftime.

Parameters

- **dt** (*datetime.datetime*) – date as basis for the URL
- **root** (*string*) – root of the url

- **fname** (*string*) – filename to use
- **subdirs** (*list, optional*) – list of strings. Each element represents a subdirectory. For example the list ['%Y', '%m'] would lead to a URL of `root/YYYY/MM/fname` or for a dt of datetime(2000,12,31) `root/2000/12/fname`

Returns url

Return type string

datedown.wget module

Interface to wget command line utility.

`datedown.wget.`**download**(*url*, *target*, *username=None*, *password=None*, *cookie_file=None*, *recursive=False*)

Download a url using wget. Retry as often as necessary and store cookies if authentification is necessary.

Parameters

- **url** (*string*) – URL to download
- **target** (*string*) – path on local filesystem where to store the downloaded file
- **username** (*string, optional*) – username
- **password** (*string, optional*) – password
- **cookie_file** (*string, optional*) – file where to store cookies
- **recursive** (*boolean, optional*) – If set then no exact filenames can be given. The data will then be downloaded recursively and stored in the target folder.

`datedown.wget.`**map_download**(*url_target*, *username=None*, *password=None*, *cookie_file=None*, *recursive=False*)

variant of the function that only takes one argument. Otherwise map_async of the multiprocessing module can not work with the function.

Parameters

- **url_target** (*list*) – first element the url, second the target string
- **username** (*string, optional*) – username
- **password** (*string, optional*) – password
- **cookie_file** (*string, optional*) – file where to store cookies
- **recursive** (*boolean, optional*) – If set then no exact filenames can be given. The data will then be downloaded recursively and stored in the target folder.

Module contents

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

d

C

D

H

M

N

P

R