py-rename

Release 2.0

Sep 11, 2020

Contents:

CHAPTER 1

Build Badges

1.1 Installation

```
pip3 install py-rename
```

1.1.1 Requirements

- Python 3.6 (or higher)

1.2 Usage

1.2.1 Command line interface

Enter `py-rename -h` for help

```
$ py-rename -h

py-rename - Python rename tool for multiple files

Usage:
```

(continues on next page)

(continued from previous page)

```
py-rename [OPTIONS] COMMAND [COMMAND-OPTIONS]

Positional arguments:
    rename              rename files based on regex pattern
    match               rename files based on regex pattern
    prefix              prefix filenames with prefix string
    postfix             postfix filenames with prefix string
    lower               convert filenames to lowercase
    replace             replace spaces in filenames to _
    camelcase           convert filenames to camel case

Options:
  -h, --help            show this help message and exit
  -V, --version         show program version number and exit
  -s, --silent          do not print output (default: False)
  -n, --dryrun          Dry run: print names of files to be renamed, but do not
                        rename (default: False)
  -f, --full            Match only full filename against pattern (default:
                        False)
```

Subcommand usage example:

```
$ py-rename rename -h

Positional arguments:
  pattern      regex pattern to match filenames
  replacement  replacement regex pattern for renamed files

Options:
  -h, --help   show this help message and exit
```

1.2.2 Examples

Example 1 - Renaming based on regex

Imagine you have some files awfully named like this:

- `ab12+Red+(0000).txt`
- `ab12+Red+(0001).txt`
- `ab12+Red+(0002).txt`

and you want to rename all of them in manner `01-Red.txt` (extracting number from the end and put it at the beginning and shortening it to 2 digits).

Step 1: Test matching pattern

Regex pattern to match those files and extract 2 digit number should be like this: *.+(00(d{2})).+*

```
$ py-rename match ".+\(00(\d{2})\).+"
('matched ab12+Red+(0000).txt',)
('matched ab12+Red+(0001).txt',)
('matched ab12+Red+(0002).txt',)
('files matched: 3',)
```

Step 2: Test replacement pattern using dryrun flag

```
$ py-rename -n rename ".+\(00(\d{2})\).+" "\1-Red.txt"
Performing DryRun: No actions will be taken
('renaming: ab12+Red+(0000).txt --> 00-Red.txt',)
('renaming: ab12+Red+(0001).txt --> 01-Red.txt',)
('renaming: ab12+Red+(0002).txt --> 02-Red.txt',)
('files matched: 3',)
```

Step 3: Actual renaming

```
$ py-rename rename ".+\(00(\d{2})\).+" "\1-Red.txt"
('renaming: ab12+Red+(0000).txt --> 00-Red.txt',)
('renaming: ab12+Red+(0001).txt --> 01-Red.txt',)
('renaming: ab12+Red+(0002).txt --> 02-Red.txt',)
('files matched: 3',)
```

Example 2 - Add prefix string or postfix string to files

Imagine you have some files named like this:

- `00-Red.txt`
- `01-Red.txt`
- `02-Red.txt`

Add prefix string:

```
$ py-rename prefix "test_"
('renaming: 00-Red.txt --> test_00-Red.txt',)
('renaming: 01-Red.txt --> test_01-Red.txt',)
('renaming: 02-Red.txt --> test_02-Red.txt',)
('files matched: 3',)
```

Add postfix string:

```
$ py-rename postfix "_test"
('renaming: 00-Red.txt --> 00-Red_test.txt',)
('renaming: 01-Red.txt --> 01-Red_test.txt',)
('renaming: 02-Red.txt --> 02-Red_test.txt',)
('files matched: 3',)
```

1.3 Code References

1.3.1 Main class

class src.py_rename.**RenameIt**(*dryrun*, *silent*, *full*)

This is a generic Rename class with several methods for various types of renaming.

Constructor args:

Parameters

- **dryrun** (*str, optional*) – just a dry run, no actual renaming performed
- **silent** (*str, optional*) – bare minimum will be printed
- **full** (*str, optional*) – apply regex on full filenames only

bulk_rename(*rename_func*, **args*)
Apply renaming function to multiple files

Parameters

- **rename_func** – specific renaming function to apply
- **args** – args for the specific renaming function

Returns None

camel_case(*filename*)
Amend filename to camel case

Parameters **filename** – str, filename

Returns True

filename_pattern_rename(*filename*, *pattern*, *replacement*, *match*)
Rename or match regex pattern, do the rename and return True or False if matched

Parameters

- **filename** – str, filename
- **pattern** – str, matching regex pattern
- **replacement** – str, replacing regex pattern, can be None
- **match** – match input based on re library

Returns True or False depending on if match or no match

lower_filename(*filename*)
Make filename all lowercase

Parameters **filename** – str, filename

Returns True

match_filename(*filename*, *pattern*, *replacement*, *full*)
Match filename function to generate matches

Parameters

- **filename** – str, filename
- **pattern** – str, matching regex pattern
- **replacement** – str, replacing regex pattern, can be None
- **full** – boolean, apply matching pattern on full filename

Returns True or False based on filename pattern rename

postfix_filename(*filename*, *postfix_str*, *include_ext=False*)
Apply postfix to filename

Parameters

- **filename** – str, filename
- **postfix_str** – str, postfix string to apply

- **include_ext** – boolean, apply postfix to filename including file extension or not

Returns True

prefix_filename (*filename*, *prefix_str*)
Apply prefix to filename

Parameters

- **filename** – str, filename
- **prefix_str** – str, prefix string to apply

Returns True

replace_space (*filename*, *fill_char='_'*)
Replace spaces with a fill character

Parameters

- **filename** – str, filename
- **fill_char** – str, char to replace spaces

Returns True

1.3.2 Argument parsing

src.main.**parse_args** (*args*)
Build argument parser function

Parameters **args** – str, arguments to parser

Returns parser args object

1.3.3 Main function

src.main.**main**()
Main function to read cli arguments and apply correct logic

Returns None

1.3.4 Unit Tests

Unit tests for this library are using pytest.

1.4 License

This library is released under the MIT License

1.4.1 MIT License

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index