Filezen

Release 1.5.3

Oct 26, 2020

Contents

Contents:

CHAPTER 1

Home

A tool which helps you organize your files by reading your file storing pattern.

If you haven't already, check out the README to get a basic overview for what this tool is about. More advanced topics and use-cases will be covered here. Start by choosing a topic you wish to learn more about from the sidebar.

For contributing, check out the GitHub repository.

CHAPTER 2

Installation

Support

- **OS Support**: Linux, Windows, Mac
- **Language Support**: Python 2.x, 3.x

2.1 Option 1: installing through pip (Recommended)

pypi package link

```
$ pip install Filezen
```

If you are behind a proxy

```
$ pip --proxy [username:password@]domain_name:port install Filezen
```

Note: If you get `command not found` then `$ sudo apt-get install python-pip` should fix that

2.2 Option 2: Installing from source (Only if you must)

```
$ git clone https://github.com/ab-anand/Filezen.git
$ cd Filezen/
$ pip install -r requirements.txt
$ python setup.py install
```

Note: If you get `permission denied` then `$ sudo python setup.py install` should fix that

CHAPTER 3

Usage

3.1 SimpleScanner

3.1.1 NOTE ON SIMPLESCANNER

- This program uses a predefined mapping

to store a particular file type from the input directory.

- Validates the folder location and moves a file

according to the mapping.

- If a file with the same name is already present in

that mapping directory, then the file won't be moved.

class filezen.simpleScanner.simplescanner.**SimpleScanner**

Bases: filezen.scanner.scanner.Scanner

This class moves the input files using a predefined mapping. Inherits some methods from Scanner.

cleanDirectory(*inputPath*, *outputPath=None*)

The main functions which takes inputPath & outputPath and validates them. Then calling the required functions in order to move the files to their respective locations

Parameters

- **outputPath** (*string*) – absolute path of the folder containing output files
- **inputPath** (*string*) – absolute path of the folder containing input files

Returns json containing list of file that were moved/not-moved

setOutputPath(*outputPath*)

set the output folder

Parameters **outputPath** (*string*) – the output folder where the files needs to be moved

3.1.2 Module contents

3.2 Advanced Scanner

3.2.1 NOTE ON ADVANCEDSCANNER

- This program uses the FREQUENCYHEAP to

maintain the most used folder location to store files of a particular type.

- Using the heap built, it move the files to the

corresponding directory.

- If a file with the same name is already present in

that mapping directory, then the file won't be moved.

class filezen.advancedScanner.advancedscanner.**AdvancedScanner**
Bases: filezen.scanner.scanner.Scanner

This class maintains a dictionary with key being the file type and value being a FREQUENCYHEAP. It then move the files according to this dictionary. Inherits some methods from Scanner.

cleanDirectory(*inputPath*, *depth=5*, *outputPath=None*)
The main functions which takes inputPath, outputPath, depth and validates them. Then calling the required functions in order to move the files to their respective locations

Parameters

- **outputPath** (*string*) – absolute path of the folder containing output files
- **depth** (*integer*) – levels of folders to scan
- **inputPath** (*string*) – absolute path of the folder containing input files

Returns json containing list of file that were moved/not-moved

setDepth(*depth*)
set the depth of scanning

Parameters **depth** (*int*) – the depth up to which scanning would be done

setOutputPath(*outputPath*)
set the output folder

Parameters **outputPath** (*string*) – the output folder where the files needs to be moved

3.2.2 Module contents

CHAPTER 4

Advanced Options

4.1 Simple Scanner

- `Simple Scanner` uses a predefined mapping of filetypes to folder `e.g. ".csv": "Documents"`.
- Based on this mapping it creates directories(only if they don't exist already) and organizes files into them as shown in the above Fig.
- Using `Simple Scanner`

```
>>> from filezen import SimpleScanner as scanner
>>> input_directory = "/home/abhinav/Downloads"
>>> output_directory = "/home/abhinav/Documents"
>>> result = scanner.cleanDirectory(input_directory, outputPath=output_directory)
>>> print(result)
'{
    "Moved": [
        "FileA.pdf",
        "FileB.txt",
        "FileC.mp4",
        "FileD.log",
        "FileB.xyz"
    ],
    "NotMoved": [
    ]
}'
```

- If no **Output Directory** is specified, then `Simple Scanner` would treat **Input Directory** as the **Output Directory** thus creating folders in the **Input Directory** itself.

4.2 Advanced Scanner

- `Advanced Scanner` maintains a `heap` for each filetype/file-extension it encounters while scanning the **Output Directory**.
- This `heap` contains all the directory addresses where a particular filetype(`e.g. "pdf"`) occurs.
- The address having the highest number of occurrence of that filetype is at the **top** of the `heap`
- With the help of this heap it finally decides the directory where a particular filetype has mostly occurred and thus moving the all such files into that directory.
- As shown in the above image, `Advanced Scanner` scans the `child` as well as `sibling directories` (at the same level).
- The level of child directories to scan is decided by `depth` parameter as shown in the example below.
- By default, `depth = 5`
- Using `Advanced Scanner`

```
>>> from filezen import AdvancedScanner as scanner
>>> input_directory = "/home/abhinav/Downloads"
>>> output_directory = "/home/abhinav/Documents"
>>> depth = 3
>>> result = scanner.cleanDirectory(input_directory, outputPath=output_directory,␣
↪depth=depth)
>>> print(result)
'{
    "Moved": [
        "FileA.pdf",
        "FileB.txt",
        "FileC.mp4",
        "FileD.log",
        "FileE.xyz"
    ],
    "NotMoved": [
    ]
}'
```

- If no **Output Directory** is specified, then `Advanced Scanner` would read the folders in the **Input Directory** itself and move accordingly.

Note: If a file with the same name is already present in the **Output Directory** then `Filezen` would ignore the file and leave it to the user. In the resulting JSON, you'll get the all such filenames which were not moved in the `NotMoved` list.

CHAPTER 5

FAQ

All FAQs will be mentioned here.

5.1 What is the significance of input folder?

Input folder is the folder which contains all your cluttered files which you want to organize.

5.2 What is the significance of output folders in Simple Scanner?

In context of Simple Scanner, output folder would be the folder inside which your scanner would create the sub-folders, e.g. Music. This is necessary as sometimes you want the organized files to be in different folder.

5.3 What is the significance of output folders in Advanced Scanner?

In context of Advanced Scanner, output folder is where you have kept you organized files. The scanner would read your output folder and understand the storing pattern. The scanner then takes files from Input folder and move them inside your output folder according to the pattern.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index