pyfilemanager

Release 1.1.0

Praneeth Namburi

Mar 04, 2024

CONTENTS

Easy to use file search and file path management in python.

CHAPTER

ONE

INSTALLATION

`pyfilemanager` can be installed through pip or conda.

```
pip install pyfilemanager

# or

conda install conda-forge::pyfilemanager
```

CHAPTER

TWO

QUICKSTART

```
from pyfilemanager import FileManager

# one-liner to retrieve videos in the canon folder
canon_videos = FileManager(r'C:\videos').add()['canon/*Camera.avi']

# or, work with tags
fm.add('canon', '*Camera.avi', include='canon')
canon_videos = fm['canon']
```

CHAPTER

THREE

USAGE

Consider the following directory structure

```
C:\videos
├── sony
│   ├── 142Camera.avi
│   ├── 143Camera.avi
│   ├── notes.txt
├── panasonic
│   ├── 151Camera.avi
│   ├── 143Camera.avi
│   ├── notes.txt
├── panasonic2
│   ├── 201Camera.avi
│   ├── 202.mp4
├── canon
│   ├── 51Camera.avi
│   ├── 40Camera.avi
│   ├── notes.txt
├── notes
│   ├── notes1.txt
│   ├── notes2.txt
```

0. Import the FileManager class.

```
from pyfilemanager import FileManager
```

1. Initialize the file manager.

```
fm = FileManager(r'C:\videos', exclude_hidden=True)
```

2. Add files based on different inclusion and exclusion criteria. Use the include parameter to keep file paths that contain **all** of the supplied strings *anywhere* in the file path. Use the exclude parameter to disregard file paths that contain **any** of the supplied string anywhere in the file path.

```
fm.add()
# add all files in the directory under the tag 'all'
# achieves the same result as fm.add('all') and fm.add('*.*')

fm.add('*.avi')
# add all files with extension .avi under the tag 'avi'
# this is short for fm.add('avi', '*.avi'), and only works for patterns that start␣
```

(continues on next page)

(continued from previous page)

```
↪with *.

fm.add('canon', '*Camera.avi', include='canon')
# include=canon means that file path must contain canon

fm.add('sony42', '*Camera.avi', include=['sony', '42'])
# file path must contain sony AND 42
# grabs one file - sony\142Camera.avi

fm.add('videos', ['*.avi', '*.mp4'])
# add multiple conditions by supplying them as a list or tuple

fm.add('panasonic', '*.avi', include='panasonic', exclude='panasonic2')
# grab videos in the panasonic folder

fm.add('notes', 'notes*.txt')
# grab all notes

fm.add('notes', 'notes*.txt', include='notes\\notes')
# grab notes from the notes folder
# this will overwrite the previous notes entry

fm.add('notes', 'notes*.txt', exclude=['sony', 'panasonic', 'panasonic2', 'canon'])
# achieves the same result as the previous line

fm.add_by_depth(max_depth=1)
# creates the tags 'files0' and 'files1' (new in v1.1)
# files0 tag will contain paths for files in the base directory (0 entries)
# files1 tag will contain paths for files in the immediate sub-directories (13␣
↪entries)
```

3. Retrieve file paths using a dict-like convention.

```
fm['canon']
# Retrieve by the tag 'canon'
# Returns canon/40Camera.avi and canon/51Camera.avi

fm['143Camera']
# When a tag is not found, retrieve file paths by an exact match to the file stem.
# Returns panasonic/143Camera.avi and sony/143Camera.avi

fm['20']
# If the key doesn't match a tag or a stem of a filename, do a loose-search to␣
↪retrieve all entries where the tag is anywhere in the full path.
# Returns panasonic2/201Camera.avi and panasonic2/202.mp4

fm['notes?.txt']
# Search for files using special chracters *, ?, !, [] specified in fnmatch
# Returns notes/notes1.txt and notes/notes2.txt
```

4. Add and retrieve in one line of code.

```
fm.add('canon', '*Camera.avi', include='canon')['canon']
# Note that the method returns the instance of the FileManager object.
```

5. Retrieve all the added keys using fm.**get_tags**()

```
fm.get_tags()
# returns a list ['canon', 'sony42', 'videos', 'panasonic', 'notes']
```

6. Retrieve paths of all the added files.

```
fm.all_files
```

7. Get a report of the number of files, and the occupied space.

```
 fm.report()
```

```
2 canon files taking up 0.000 MB
1 sony42 files taking up 0.000 MB
8 videos files taking up 0.000 MB
2 panasonic files taking up 0.000 MB
5 txt files taking up 0.000 MB
```

CHAPTER

FOUR

LICENSE

Distributed under the MIT License. See `LICENSE` for more information.

CHAPTER

FIVE

CONTACT

Praneeth Namburi

Project Link: https://github.com/praneethnamburi/pyfilemanager

CHAPTER

SIX

ACKNOWLEDGMENTS

This tool was developed as part of the ImmersionToolbox initiative at the MIT.nano Immersion Lab. Thanks to NCSOFT for supporting this initiative.

6.1 API Reference

Easy to use file search and file path management.

FileManager class initializes file path management in the directory of interest. *FileManager.add()* is used to tag a set of file paths filtered based on different inclusion and exclusion criteria. *FileManager.__getitem__()* is used to retrieve file paths of interest based on a tag, filename, or pattern. *find()* is the core function for finding files, and it is based on *os.walk* and *fnmatch*.

class pyfilemanager.**FileManager**(*base_dir: str*, *exclude_hidden: bool = True*)

Easy to use file search and file path management. Add files using different inclusion and exclusion criteria under a 'tag'. Provides dictionary-like access to file-paths where the tags serve as keys. Useful for managing files in not-so-obviously organized folders.

Parameters

- **base_dir** (*str*) – base directory for file search
- **exclude_hidden** (*bool, optional*) – excludes hidden files when True. Defaults to True.

Variables

- **base_dir** (*str*) – base directory for file search
- **_files** (*dict*) – {Tag: List of file paths}
- **_filters** (*dict*) – {Tag: pattern list}
- **_inclusions** (*dict*) – {Tag: inclusion criteria}
- **_exclusions** (*dict*) – {Tag: exclusion criteria}

add(*tag: str = 'all'*, *pattern_list: str | list[str] = None*, *include: str | list[str] = None*, *exclude: str | list[str] = None*, *exclude_hidden: bool = None*) → *FileManager*

Add files based on different inclusion and exclusion criteria. Call this method without any arguments to work with all the files in the directory using *FileManager.__getitem__*. Note that if a tag already exists, it will get overwritten with the new

Examples

Add files that match the pattern *Camera*.avi under the tag *video*

```
fm.add('video', '*Camera*.avi')
```

Add all files under the tag *all* (special case)

```
fm = FileManager(r'C:\videos').add()
```

Parameters

- **tag** (*str, optional*) – e.g. 'video_files'. Defaults to all, meaning add all files in the directory recursively.
- **pattern_list** (*Union[str,list], optional*) – e.g. '*.avi', [*'.avi', '*.mp4']. Defaults to *.
- **include** (*Union[str,list], optional*) – Keep file paths that contain **all** of the supplied strings *anywhere* in the file path. Defaults to None.
- **exclude** (*Union[str,list], optional*) – Disregard file paths that contain **any** of the supplied string anywhere in the file path. Defaults to None.
- **exclude_hidden** (*bool, optional*) – Set the state for excluding hidden files. Defaults to the value of _exclude_hidden attribute, which defaults to True.

Returns
Returns self. Useful for chaining commands.

Return type
FileManager

add_by_depth(*max_depth: int = 0*, *exclude_hidden: bool = None*, *include_directories: bool = False*)

Add files and directories by their depth. Tags of name files0, and directories0 will be created for files and directories at depth0.

Parameters

- **max_depth** (*int, optional*) – Maximum depth for the search. Defaults to 0, adding the top level contents only.
- **exclude_hidden** (*bool, optional*) – Include or exclude hidden files and directories. Defaults to the value of self._exclude_hidden, which defaults to True.
- **include_directories** (*bool, optional*) – When set to true, one tag will be created for directories, and one for files at each depth. When set to False, only the files tag will be created at each depth. Defaults to False.

remove(*tag: str*) → None

Remove file paths stored under the given tag.

Parameters
tag (*str*) – A tag created when using the add method.

Raises
ValueError – If an unknown tag is supplied.

__getitem__(*key: str*) → list

Retrieve file paths based on -

(0) *FileManager.filter* method if key has special chacters such as *, ?, !, []

(1) tag

(2) exact match for the 'stem' of the file

(3) key is anywhere in the path

Try (0) if there are special characters in *key*. If not, try (2) only if (1) doesn't return any results, and try (3) only if (2) doesn't return any results.

Parameters
key (*str*) – Either a tag, filename, or partial match.

Returns
List of file paths.

Return type
list

filter(*pattern: str*) → list

Filter self.all_files using *fnmatch.filter*.

Parameters
pattern (*str*) – e.g. *.avi, *notes?.txt

Returns
List of file paths.

Return type
list

get_tags() → list

Return a list of tags created using the add method.

Returns
List of tags.

Return type
list

property all_files: list

Return a list of all files managed by the filemanager. Remove duplicates.

Returns
List of file paths

Return type
list

report(*units: str = 'MB'*) → None

Print a report summarizing the size occupied by files under each tag.

Parameters
units (*str, optional*) – One of ('B', 'KB', 'MB', 'GB', 'TB). B is for bytes. Defaults to 'MB'.

pyfilemanager.**find**(*pattern: str*, *path: str = None*, *exclude_hidden: bool = True*) → list

Core function for finding files based on `os.walk` and `fnmatch`.

Example

```
find('*.txt', r'C:\videos')
```

Parameters

- **pattern** (*str*) – Input for fnmatch.
- **path** (*str, optional*) – Search for files in this path. Defaults to the results of os.getcwd().
- **exclude_hidden** (*bool, optional*) – Whether to include filenames of hidden files. Defaults to True.

Returns
List of file names.

Return type
list

6.2 Demo

https://youtu.be/ECf5KjdngOU

6.3 Change Log

All notable changes to this project will be documented in this file.

6.3.1 [1.1.0] - 2024-02-22

Added

Depth-based search support for files and directories. Useful for working with files in the top level directory.

```
FileManager(base_dir).add_by_depth()['*annotations*.json']
# To retrieve annotation json files in base_dir and not with any of the files in the sub-
↪directories.

FileManager(base_dir).add()['*annotations*.json']
# To retrieve annotation json file paths in the base directory and all the sub-
↪directories.
```

Changed

1. When `exclude_hidden` is set to `True` (default), a folder called *#recycle* will now be ignored. This is to exclude the contents of the recycle bin on network attached storage devices such as a synology NAS.
2. Refactored the core function `pyfilemanager.find`

6.3.2 [1.0.0] - 2024-01-26

First major release after thorough testing, 100% coverage, and formatting.

Added

More flexibility in the FileManager.add method. When called without any arguments, all the files in the base directory are stored in the FileManager object.

6.3.3 [0.1.1] - 2024-01-21

Initial release

PYTHON MODULE INDEX

INDEX

Symbols

A

F

G

M

P

R