Søren Madsen
Part of my descriptive guides collection, Version 1.0

# SortFilesToFolders

*Python 3 based file sorting*

Table of contents

# Intro

This document describes the Python based script SortFilesToFolders.

## 1.1 Disclaimer

The script has only been sporadically tested on:

- Various versions of: Microsoft Windows 10.
- It has been running on both python 2.7x and 3.x versions.

  Current state: It should run without warnings on Python 3.10.

In other words it has NOT been tested on a full blown Continuous Integration server with 100% code coverage. The script should ***NOT*** be considered production ready software and it was developed for my personal setup on my own PC. Some of the files contain functions that are untested.

The script does run and works quite well. It has been tested to a minor extent by assert functions, and to a greater extend by manually executing the script on live test-data.

As always, please test this software on a set of test data before you use it on your own precious data, to be sure you have set it up correctly!

If you use the ‘-m‘ flag when parsing arguments to the script for instance by using the file organizePictures.bat the files will be *moved* into the output folder and your source data will “dissapear”. By avoiding the usage of ‘-m’ your source data should be intact after executing the script because a copy has been used instead of a move function.

Feel free to use this script/software at your own risk.

## 1.2 Basic requirements

The script should be able to sort individual folders and files into output folders organized by the exif data/dates stored in the files. If no exif data is found the script can use the operating system data instead. If no date data is found a default date is used as output-name generation.

The code should be generalized and wrapped in easy to use classes and functions in a simple object oriented programming style. There haven’t been defined any requirements regarding programming patterns and the code is simplistic and abit sequential although object object oriented.

## 1.3 Limitations and shortcomings

The script is a console application so no graphical user interface is provided.

The code base has been developed using an adhoc approach leaving some process improvements to be desired :-). That is a luxury one has when coding own projects for fun, but a better analysis and development process would minimize a lot of "rushed code errors" and thereby minimize the time spent debugging the code afterwards.

The project does not handle thread synchronization problems. So running multiple scripts at the same time is out of the scope of this implementation.

### 1.4 Structure

The script consists of several python files, a bat file and an attempt to make a small generalized python python library called UserPackages.

The structure of the script:

```
|-- .\UserPackages
    |-- \GenericFunctions
        |-- __init__.py
        |-- file_path.py
        |-- scheduler.py
        |-- time.py
        |-- utility.py
|-- .\SortFilesToFolders
    |-- .\Documentation
        |-- HowTo_Description.pdf
    |-- .\SortTheseFiles
    |-- Install required library.txt
    |-- organizePictures.bat
    |-- sortFilesToFolders.py
```

**User library base folder: Userpackages**

This folder is the user library folder. It was thought of as an attempt to make a library with generalized code to be used by multiple python projects. Some of the functions might be abit too tightly coupled to the SortFilesToFolders.py script to be categorized as generalized code :-). But consider it an example of how to make your own personal python library instead of putting everything in the official Python class path.

**File: __init__.py**

The file is used to mark a folder on disk as a Python package directory. That ensures that Python searches this folder for submodules when the import statement is used in the script.

**Module sub folder: GenericFunctions**

The folder containing some generic library classes and functions.

**File: __init__.py**

The file is used to mark a folder on disk as a Python package directory. That ensures that Python searches this folder for sub modules when the import statement is used in the script.

https://docs.python.org/3/reference/import.html#regular-packages

**File: file_path.py**

This file contains functions the handle file path operations.

**File: scheduler.py**

This file has not been implemented. It could be the basis of some kind of synchronization or thread handling system.

**File: time.py**

This file handles everything time related:

- Current time

- Timers

- DateTime formatting

**File: utility.py**

This file contains code regarding console output, exit function etc.

**Script base folder: SortFilesToFolders**

This folder is the python script entry point. In here lies the main python Class, that runs the script.

**Documentation folder: Documentation**

Contains this file.

**Source folder: SortTheseFiles**

Default source folder.

**File: Install required library.txt**

Text file describing libraries required by the script and how to install them.

**File: organizePictures.bat**

A bat file that demonstrates how to start the script with several command line argument combinations.

**File: Install required sortFilesToFolders.py**

The main script entry point.

## 1.5 Installation tips

The script requires the following software to work:
- Microsoft Windows
- Python 3.X installation. Tested on Python 3.10.

- Add python.exe to the path environment variable.
- Run organizePictures.bat