03-organize_photos_by_date-v01

Photography Workflow Documentation

Yahya Nazer - ChatBizDB.Com

2026.06.18

Table of contents

Organize Photos by Date — Documentation

1. Purpose

This Python program organizes photo files into subfolders based on their date.

It first tries to read the EXIF **Date Taken** from each image. If that date is not available, it uses the file's modified date from the operating system.

2. What the Program Does

1. Asks the user to select an input folder using a GUI.
2. Asks the user to select an output folder using a GUI.
3. Asks whether to copy or move the files.
4. Searches the input folder and all subfolders.
5. Detects supported photo files.
6. Reads the photo date.
7. Creates date-based folders.
8. Copies or moves files into the correct date folder.
9. Shows a summary when finished.

3. Default Folder Format

The default output folder format is:

```
YYYY-MM-DD
```

Example:

```
2026-05-14
2026-05-15
2026-05-16
```

Inside each folder, the matching photos are copied or moved.

4. Supported File Types

The script supports common photography formats:

```
.jpg
.jpeg
.png
.tif
.tiff
.heic
.cr2
.cr3
.nef
.arw
.raf
.dng
.3fr
.fff
```

5. Required Package

The program requires Pillow.

Install it with:

```
pip install pillow
```

6. Main Functions

`select_folder(title)`

Opens a GUI folder picker and returns the selected folder.

`ask_copy_or_move()`

Asks whether the user wants to copy files or move files.

- Yes = Copy
- No = Move

`get_exif_date_taken(file_path)`

Reads EXIF metadata from an image and tries to find:

- `DateTimeOriginal`
- `DateTimeDigitized`
- `DateTime`

`get_file_modified_date(file_path)`

Uses the operating system modified date when EXIF date is not available.

`get_photo_date(file_path)`

Decides which date to use:

1. EXIF date if available
2. Modified date if EXIF is missing

`create_date_folder(output_folder, photo_date, folder_style)`

Creates the destination folder based on the photo date.

Supported styles in the code:

```
YYYY
YYYY-MM
YYYY-MM-DD
YYYY/MM/DD
```

The default is:

```
YYYY-MM-DD
```

`get_unique_destination_path(destination_folder, file_name)`

Prevents overwriting files.

If a file already exists, it creates names like:

```
IMG_1001_001.jpg
IMG_1001_002.jpg
IMG_1001_003.jpg
```

`is_photo_file(file_name)`

Checks if the file extension is supported.

It also skips macOS hidden files such as:

```
._filename.jpg
```

`organize_photos(input_folder, output_folder, action_type, folder_style)`

Main processing function.

It scans files, gets dates, creates folders, and copies or moves the photos.

`main()`

Controls the full workflow.

7. Inputs

The program asks for:

1. Input folder containing photos
2. Output folder for organized photos

3. Copy or move selection

8. Outputs

The program creates date-based folders and places photos into them.

It also displays a final summary:

- Total files scanned
- Photos processed
- Files skipped
- Errors

9. Safety Notes

Use **Copy** first when testing.

Only use **Move** after confirming the output structure is correct.

10. Recommended Project Folder

Example:

```
Photo-Organizer
│
├── A-Data
│   └── Original-Photos
│
├── B-Engines
│   └── organize_photos_by_date.py
│
└── C-Reports
    └── Organized-Photos
```

11. How to Run

1. Open VS Code.
2. Open the folder containing the script.
3. Install Pillow if needed:

```
pip install pillow
```

4. Run:

```
python organize_photos_by_date.py
```

5. Select the input folder.
6. Select the output folder.
7. Choose Copy or Move.