04-FolderScan-SelectDelete-v01

Photography Workflow Documentation

Yahya Nazer - ChatBizDB.Com

2026.06.18

Table of contents

Overview

FolderScan-SelectDelete (`FolderScan-SelectDelete-V03.py`) is a Python desktop utility that scans any folder tree, exports a full inventory to a timestamped CSV file, lets you mark items with action codes directly in Excel or LibreOffice, and then executes those actions through a colour-coded review GUI.

The tool uses only the Python standard library — `tkinter`, `os`, `csv`, `shutil`, `re`, `platform`, `subprocess`, `pathlib`, `datetime` — and runs on **Windows**, **macOS**, and **Linux** without any additional packages.

Design Principles

- **Non-destructive by default.** Nothing changes until the user explicitly confirms inside the Phase 3 review window.
- **CSV as the control file.** The scan output is a plain `.csv` that opens in any spreadsheet editor. The user marks actions in column A and saves — no custom GUI editor required.
- **Single combined launcher.** The launcher is one split-panel window: workflow steps on the left, full command-code reference on the right. Everything you need is visible without opening a second window.
- **Audit trail.** Every Phase 3 run produces a timestamped `ActionLog-yyyy-mm-dd--hh-mm.csv` recording every action and its outcome.
- **Date-safe naming.** Folder and file names that look like dates are written to the CSV with a leading apostrophe (`'`) so Excel does not auto-convert them to date values.

The Launcher Window

Running the script with `python FolderScan-SelectDelete-V03.py` opens a single window that fills the full height of the screen. It is divided into two panels side by side.

Left Panel — Workflow Steps

Three step blocks (green / blue / red) describe the process in plain language, followed by the four phase buttons:

Button	Colour	Calls
Phase 1 — Scan Folder → CSV	Green	`phase1_scan()`
Phase 2 — Open CSV for Marking	Blue	`phase2_open_csv()`
Phase 3 — Review & Execute Actions	Red	`phase3_execute()`
Run All Three Phases in Sequence	Purple	phases 1 → 2 → 3

Each button hides the launcher (`root.withdraw()`), runs the selected phase, and then restores it (`root.deiconify()`) when the phase completes or is cancelled.

Right Panel — Command Codes

A permanent reference panel showing a large colour-coded block for each command. Keep the launcher open beside Excel during Phase 2 so you can consult it without switching windows.

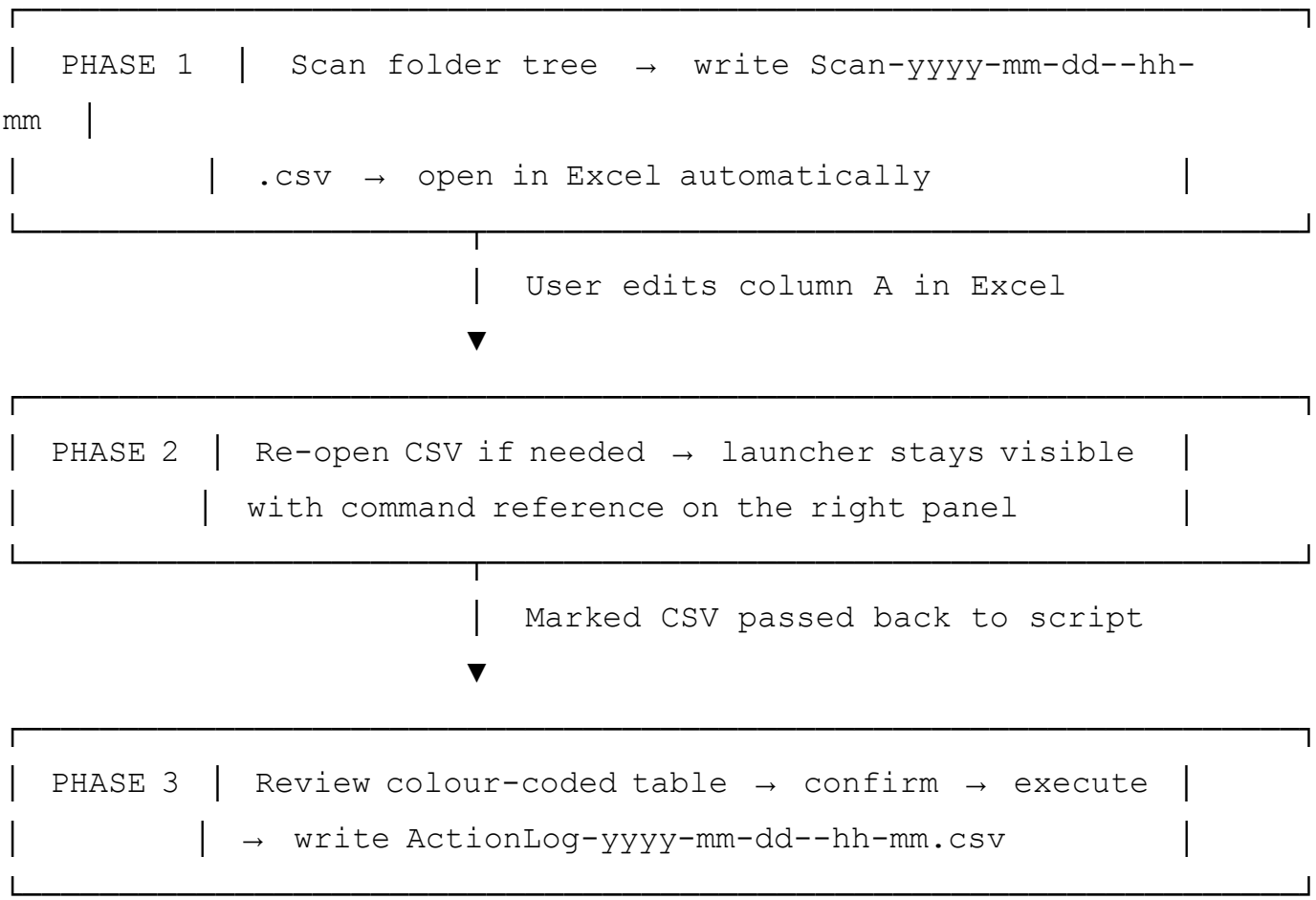
Code	Panel colour	Applies to	Meaning
`R`	Amber	FOLDER only	Delete all FILES inside; folder kept empty
`D`	Red	FILE or FOLDER	Permanently delete
`M`	Green	FILE or FOLDER	Archive copy — original kept
`X`	Purple	FILE or FOLDER	Archive move — original removed

Window Sizing

```
screen_h = root.winfo_screenheight()
screen_w = root.winfo_screenwidth()
win_w    = min(1060, screen_w)
root.geometry(f"{win_w}x{screen_h}+0+0")
```

The window is placed at the top-left corner at full screen height. It is resizable so you can adjust it on any monitor layout.

Workflow at a Glance



Configuration Flags

Edit these four variables at the top of the script before running:

Flag	Default	Description
`SCAN_RECURSIVE`	`True`	`True` = walk all subfolders; `False` = top level only
`INCLUDE_FOLDERS_IN_CSV`	`True`	`True` = include subfolder rows in CSV
`CSV_PREFIX`	`"Scan"`	Prefix of the output filename

Flag	Default	Description
`AUTO_OPEN_CSV`	`True`	`True` = open CSV in Excel automatically after Phase 1

Phase 1 — Scan and Export CSV

What It Does

Phase 1 walks the chosen root folder, records every file and subfolder it finds, and writes a structured CSV that opens automatically in Excel.

Step-by-Step

1. A **folder picker** dialog opens; the user selects the root folder.
2. `os.walk()` traverses the entire tree (or top level only if `SCAN_RECURSIVE = False`).
3. For every **subfolder** — excluding the root itself — a `FOLDER` row is appended.
4. For every **file** a `FILE` row is appended.
5. All names pass through `safe_name()` before being written. Date-like names are prefixed with `'` to prevent Excel auto-conversion.
6. The CSV is saved inside the scanned folder:

   `Scan-yyyy-mm-dd--hh-mm.csv`

7. If `AUTO_OPEN_CSV = True` the file opens in the default application.
8. A summary dialog reports total files, total folders, and total size.

CSV Structure

Zone	Row(s)	Content
Info block	1 – 10	Key-value metadata rows; first cell starts with #
Separator	11	Blank row
Headers	12	Column names
Data	13 +	One row per file or folder found

Info Block (Rows 1 – 10)

Key	Example value
`#Script`	`FolderScan-SelectDelete-V03.py`
`#Version`	`3.0`
`#Author`	`Control Designer LLC`
`#Scan_Date`	`2025-10-12 14:30:00`
`#Root_Folder`	`D:\Projects\Archive`
`#Total_Files`	`1842`
`#Total_Folders`	`47`
`#Total_Size`	`3.21 GB`
`#Recursive`	`Yes`
`#Commands`	`R=remove files in folder | D=delete | M=archive copy | X=archive move`

Row 11 is a blank separator row. Phase 3 skips every row whose first cell starts with # or is empty, so these rows are never treated as data.

Data Columns

Col	Field	Description
A	`Select`	**Leave blank.** Enter a command code here in Phase 2.
B	`Type`	`FILE` or `FOLDER`
C	`Name`	Filename or folder name (date-like names prefixed with `'`)
D	`Full_Path`	Absolute path used by Phase 3 to perform the action
E	`Size_Bytes`	Raw byte count — numeric, sortable
F	`Size_Human`	Human-readable size: `KB`, `MB`, `GB`
G	`Date_Modified`	Last modification timestamp `yyyy-mm-dd hh:mm:ss`
H	`Date_Created`	Creation timestamp
I	`Extension`	File extension e.g. `.jpg`, `.pdf` — blank for folders
J	`Depth`	Folder depth relative to root (`0` = first level below root)

Date-Safe Naming

Excel auto-converts date-like text into date serial numbers. The script prevents this with two functions:

`is_date_like(name)` tests against seven compiled regular expressions:

Pattern	Example
`yyyy-mm-dd`	`2024-01-15`
`dd/mm/yyyy`	`15/01/2024`
`yyyy-mm`	`2024-01`
`yyyymmdd`	`20240115`

Pattern	Example
`dd-mm-yy`	`15-01-24`
`Mon-yyyy`	`Jan-2024`
`yyyy-Mon`	`2024-Jan`

`safe_name(name)` prefixes a matching name with `'`:

```
"2024-01-15"  →  "'2024-01-15"
```

Excel renders the apostrophe as a hidden text-prefix marker. Phase 3 strips it before using the path.

Phase 2 — Mark Items in Excel

What It Does

Phase 2 is the only phase performed **outside** the script. The Phase 2 button re-opens the CSV (if it was closed) and immediately restores the launcher so the right-panel command reference stays visible while you edit in Excel.

Step-by-Step

1. Click **[Phase 2 — Open CSV for Marking]**. The CSV opens in Excel or LibreOffice and the launcher window comes back to the foreground.
2. Keep the launcher beside Excel as a command reference.
3. In **column A**, type one of the four codes next to each item you want to act on.
4. Save the file as **`.csv`** (not `.xlsx`).
5. Return to the launcher and click **Phase 3**.

Command Codes

Code	Applies to	Action
`R`	FOLDER only	Every FILE inside the folder is deleted. The folder itself and any sub-folders are kept intact.
`D`	FILE or FOLDER	Permanently deleted. A folder is removed with all its contents.
`M`	FILE or FOLDER	A sibling folder `<name>-yyyy-mm-dd--hh-mm` is created. The item is **copied** into it. Original unchanged.
`X`	FILE or FOLDER	A sibling folder `<name>-yyyy-mm-dd--hh-mm` is created. The item is **moved** into it. Original removed.

Upper or lower case is accepted (`r = R`, `d = D`, `m = M`, `x = X`). Any other value in column A is silently ignored by Phase 3.

Tips for Working in Excel

- **Sort column B** (`Type`) to group all `FOLDER` rows together.
- **Sort column F** (`Size_Human`) descending to surface the largest files.
- **Sort column I** (`Extension`) to batch-mark all `.tmp`, `.log`, or `.bak` files at once with `D`.
- **Sort column J** (`Depth`) to target only top-level items.
- Use `M` before `D` when you want to keep a snapshot before deleting.
- `R` is ideal for emptying cache or temp folders while preserving the folder structure.

Phase 3 — Review and Execute

What It Does

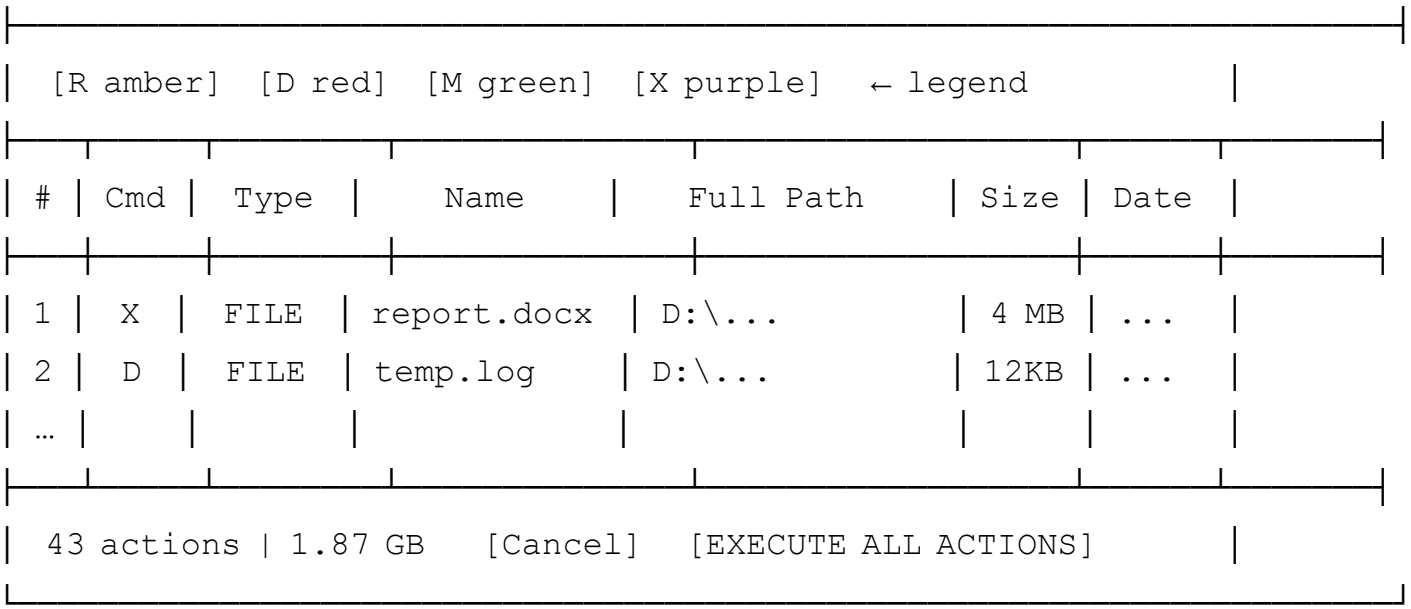
Phase 3 reads the marked CSV, opens a colour-coded review window for verification, and executes every action after a final confirmation dialog. An audit log is always written regardless of outcome.

Step-by-Step

1. A **file picker** dialog asks the user to select the marked CSV.
2. The script reads every row, automatically skipping info-block (`#`) rows and the blank separator.
3. Rows where column A is `R`, `D`, `M`, or `X` are collected as *marked rows*; the command is normalised to upper case.
4. If no marked rows are found, an informative dialog explains what to do and the phase exits without changing anything.
5. The **review window** opens.
6. The user clicks **[EXECUTE ALL ACTIONS]**.
7. A confirmation dialog lists up to 30 items (remainder summarised as a count).
8. On confirmation each action is executed in turn.
9. `ActionLog-yyyy-mm-dd--hh-mm.csv` is written.
10. A final summary dialog reports total successes and any errors.

Review Window Layout

```
┌──────────────────────────────────────────────────────────────┐
│ REVIEW BEFORE EXECUTION | 43 items | R:1 D:12 M:0 X:30       │
│  | Size affected: 1.87 GB                                    │
├──────────────────────────────────────────────────────────────┤
│ CSV: D:\Projects\Archive\Scan-2025-10-12--14-30.csv          │
```



Review Window Colour Coding

Colour	Hex	Command
Amber	`#fde68a`	`R` — Remove files in folder
Red	`#fecaca`	`D` — Delete
Green	`#bbf7d0`	`M` — Archive copy
Purple	`#e9d5ff`	`X` — Archive move

Action Execution Details

R — Remove all files inside a folder

```
for root_d, dirs, files in os.walk(path):
    for f in files:
        os.remove(os.path.join(root_d, f))
```

Walks the entire sub-tree and calls `os.remove()` on every file. Sub-folders and the target folder itself are untouched. Applying `R` to a `FILE` row raises `ValueError`, which is caught, logged as `ERROR`, and execution continues with the next row.

D — Delete permanently

```
# FILE:
os.remove(path)

# FOLDER:
shutil.rmtree(path)
```

Files are removed with `os.remove()`. Folders use `shutil.rmtree()`, which deletes the folder and **all its contents** recursively. This action is irreversible.

M — Archive copy (original kept)

```
archive_dir = os.path.join(parent_dir, f"{name}-{timestamp}")
os.makedirs(archive_dir, exist_ok=True)

shutil.copy2(path, os.path.join(archive_dir, name))    # FILE
shutil.copytree(path, os.path.join(archive_dir, name)) # FOLDER
```

Creates `<name>-yyyy-mm-dd--hh-mm` alongside the original and copies the item into it. `shutil.copy2` preserves metadata. `shutil.copytree` copies the entire folder tree. The **original is unchanged**.

X — Archive move (original removed)

```
archive_dir = os.path.join(parent_dir, f"{name}-{timestamp}")
os.makedirs(archive_dir, exist_ok=True)
shutil.move(path, os.path.join(archive_dir, name))
```

Creates the same timestamped sibling folder as `M`, then moves the item into it with `shutil.move()`. The **original path no longer exists** after the operation.

Leading Apostrophe Handling

Before any action, the script strips the `'` prefix added by `safe_name()`:

```
if name.startswith("'"):
    name = name[1:]
```

This ensures the correct file-system path is used when the name was a date-like string.

Action Log

Every Phase 3 run writes:

`ActionLog-yyyy-mm-dd--hh-mm.csv`

in the same folder as the original scan CSV.

Column	Description
`Command`	`R`, `D`, `M`, or `X`
`Status`	`OK` or `ERROR`
`Type`	`FILE` or `FOLDER`
`Name`	Item name (apostrophe stripped)
`Full_Path`	Original absolute path
`Detail`	Success note (e.g. `Removed 14 file(s); folder kept`) or error message
`Executed_At`	Timestamp of execution `yyyy-mm-dd hh:mm:ss`

Function Reference

Shared Helpers

Function	Signature	Description
`human_size`	`(num_bytes) -> str`	Bytes to `KB` / `MB` / `GB` string
`is_date_like`	`(name) -> bool`	`True` if name matches a date-like regex pattern
`safe_name`	`(name) -> str`	Prefix date-like names with `'` for Excel safety
`select_root_folder`	`() -> str|None`	GUI folder picker dialog
`select_csv_file`	`(title) -> str|None`	GUI CSV file picker dialog
`open_csv_in_default_app`	`(csv_path)`	Cross-platform file open (Windows / macOS / Linux)
`cmd_color`	`(cmd) -> str`	Hex background colour for R / D / M / X
`add_copyright`	`(parent, bg, fg)`	Append copyright label to any Tk window

Phase 1

Function	Description
`scan_folder_tree(root_path)`	`os.walk` traversal; returns `(rows, total_files, total_folders, total_bytes)`
`write_csv_with_info(rows, root_path, …)`	Write info block + blank row + headers + data rows
`show_scan_summary(total_files, …)`	Summary messagebox after scan
`phase1_scan()`	Entry point; returns CSV path or `None` if aborted

Phase 2

Function	Description
`show_instructions_window(restore_launcher)`	Standalone instruction window with Back and OK buttons
`phase2_open_csv(csv_path, restore_launcher)`	Open CSV in default app; call `restore_launcher()` to bring back the launcher

Phase 3

Function	Description
`read_marked_rows(csv_path)`	Returns `(all_rows, marked_rows)` — skips `#` rows and blank separator
`execute_actions(marked_rows, parent_window)`	Executes R / D / M / X per row; returns `(successes, errors, log_entries)`
`write_action_log(log_entries, csv_path)`	Writes `ActionLog-...csv` next to the scan CSV
`show_action_summary(ok, errors, log_path, parent)`	Final summary dialog
`build_review_gui(marked_rows, csv_path)`	Scrollable review window; entry point to execution
`phase3_execute()`	Entry point

Launcher

Function	Description
`main()`	Full-height split-panel launcher; wires all phase buttons and the combined reference panel

Running the Script

Prerequisites

Requirement	Notes
Python	3.8 or later
External packages	None — standard library only
tkinter	Included with Python on Windows and macOS. Linux: `sudo apt install python3-tk`

Command

```
python FolderScan-SelectDelete-V03.py
```

The launcher opens at full screen height immediately.

Typical Session

```
1.  Click  [Phase 1 — Scan Folder → CSV]
    Select:  D:\Projects\Archive
    → Saved:  D:\Projects\Archive\Scan-2025-10-12--14-30.csv
    → Opens in Excel automatically

2.  In Excel:
    Sort column I (Extension) → mark all .tmp files D
    Sort column F (Size_Human) desc → mark large old files X
    Type R on the "Temp" folder row
    Save as CSV
```

```
3.  Click  [Phase 3 — Review & Execute Actions]
    Select:  Scan-2025-10-12--14-30.csv
    Review window: 43 items  (D:12  R:1  X:30  M:0 | 1.87 GB)
    Click  [EXECUTE ALL ACTIONS]  →  Confirm  →  OK
    → 43 actions completed
    → ActionLog-2025-10-12--14-31.csv saved
```

Safety and Limitations

Topic	Notes
Undo	`D` is permanent. Use `M` or `X` for a recoverable archive.
Locked files	On Windows, open files cannot be deleted. The error is caught, written to the ActionLog, and execution continues.
Symbolic links	`shutil.rmtree` on a symlink removes the link, not the target directory.
Network paths	UNC paths (`\\server\share\...`) are supported but scanning may be slow.
R on a FILE row	Raises `ValueError` — caught and logged as `ERROR`; execution continues.
Excel BOM	CSV written with `utf-8-sig` (UTF-8 BOM) for correct Excel display on Windows.
Other column A values	Non-command values in column A are silently ignored. You may add notes freely.
Archive folder collision	`os.makedirs(exist_ok=True)` reuses an existing archive folder if the same timestamp is present.

Copyright

This tool is provided for internal and personal productivity use. The authors accept no liability for unintended data loss. Always review the Phase 3 window carefully before confirming any destructive action.