perplexity

Give me another assignment involving practical cases of file handling in python. Remember the course focuses on IT Automation. Find a common problem that can be automated using the concepts mentioned before.

Here is a practical assignment aligned with IT automation, focusing on automating file organization—a common real-world automation need in many IT environments. This task helps consolidate your file handling knowledge and demonstrates a direct benefit of Python scripting for system admins and support roles.

Assignment: Automated Downloads Folder Organizer

Objective:
Develop a Python script to automatically organize files in a "Downloads" directory into subfolders by type (such as Documents, Images, Audio, Video, Archives, and Others).

Scenario

A typical problem in IT workplaces is cluttered downloads or transfer directories, where files accumulate unsorted. Manually sorting hundreds of files is time-consuming and error-prone. Automating this task improves organization, saves time, and reduces mistakes[1] [2] [3] [4].

Tasks

1. **Setup**
    - Your script should work with a specified target directory (default: the user's "Downloads" folder).
    - Use only **relative paths** inside your script logic.
2. **Directory Structure**
    - For each run, ensure subfolders named “Documents”, “Images”, “Audio”, “Video”, “Archives”, and “Others” exist, creating them if they don't, with `os` and `os.path`.
3. **File Detection & Categorization**
    - Scan the target directory for files.
    - Sort files based on their extensions (e.g., `.pdf`, `.jpg`, `.mp3`, `.mp4`, `.zip`), moving them to the appropriate subfolder.
    - Place files with unknown or uncategorized extensions into the "Others" folder.
    - Only organize/move actual files, not folders.

4. **File Move & Name Conflict Handling**
    - If a destination filename already exists, append a numbered suffix to avoid overwriting (e.g., `report(1).pdf`).
5. **Summary Output**
    - After organizing, print (or write to `organizer_log.txt`):
        - Number of files moved per category.
        - List of files and where they were placed.

Requirements

- Use the `os` and `os.path` libraries for all file and directory operations.
- Include error handling for permissions, missing files, and name conflicts.
- Your script should not require any user input after launch (automated).

Example: Before & After

Downloads folder (Before):

```
Downloads/
    report.pdf
    sunset.jpg
    song.mp3
    video.mp4
    archive.zip
    old_version.pdf
    randomfile.xyz
```

Downloads folder (After running script):

```
Downloads/
    Documents/
        report.pdf
        old_version.pdf
    Images/
        sunset.jpg
    Audio/
        song.mp3
    Video/
        video.mp4
    Archives/
        archive.zip
    Others/
        randomfile.xyz
    organizer_log.txt
```

Deliverables:

- The Python script you created.
- Screenshot or list showing your "Downloads" directory before and after running the automation.

Optional bonus: Make your script configurable via a settings file, or schedule it using Task Scheduler (Windows) or cron (Linux).

This assignment is highly practical and mirrors many common IT automation scenarios, such as keeping shared folders tidy, improving backup management, and minimizing manual interventions[1] [2] [3] [4]. If you'd like code hints or further extension ideas, just ask!

⁂

1. https://www.index.dev/blog/python-automation-scripts
2. https://consoleflare.com/blog/files-oragnizer-with-python-in-10-steps/
3. https://www.geeksforgeeks.org/python/automating-file-movement-on-your-system/
4. https://www.datacamp.com/tutorial/python-automation