codealpha4

December 7, 2024

1 Task Automation with Python Scripts

1.0.1 Identify a repetitive task in your workflow and create

1.0.2 Python scripts to automate it. This could include tasks

1.0.3 like file organization, data cleaning, or system

1.0.4 maintenance

2 *File Organization*

[25]:
```python
import os

def create_sample_files(folder_path):
    """Create a sample directory with files for demonstration."""
    # Ensure the folder exists
    os.makedirs(folder_path, exist_ok=True)

    # Define sample files to create
    sample_files = [
        "report1.pdf",
        "image1.jpg",
        "code1.py",
        "notes1.txt",
        "presentation1.pptx",
        "spreadsheet1.xlsx",
        "image2.png",
        "report2.pdf",
        "code2.py",
    ]

    # Create the files in the folder
    for file_name in sample_files:
        file_path = os.path.join(folder_path, file_name)
        with open(file_path, "w") as file:
            file.write(f"Sample content for {file_name}")
    print("Sample files created successfully.\n")
```

```python
def organize_files_by_extension(folder_path):
    """Organize files in the given folder by their extensions."""
    print("Organizing files by extensions...\n")

    for file_name in os.listdir(folder_path):
        file_path = os.path.join(folder_path, file_name)

        # Skip directories
        if not os.path.isfile(file_path):
            continue

        # Get file extension and create a folder
        _, extension = os.path.splitext(file_name)
        extension = extension.lstrip(".").lower()

        # Handle files without extensions
        if not extension:
            extension = "no_extension"

        extension_folder = os.path.join(folder_path, extension)
        os.makedirs(extension_folder, exist_ok=True)

        # Prepare the new file path
        new_file_path = os.path.join(extension_folder, file_name)

        # Handle name conflicts
        if os.path.exists(new_file_path):
            base_name, ext = os.path.splitext(file_name)
            counter = 1
            while os.path.exists(new_file_path):
                new_file_name = f"{base_name}_{counter}{ext}"
                new_file_path = os.path.join(extension_folder, new_file_name)
                counter += 1

        # Move the file
        os.rename(file_path, new_file_path)
        print(f"Moved: {file_name} -> {new_file_path}")

    print("\nFiles have been organized successfully.\n")


def display_folder_contents(folder_path):
    """Display the contents of the folder."""
    print("Folder structure after organization:\n")
    for root, dirs, files in os.walk(folder_path):
        print(f"Folder: {root}")
```

```
        for file in files:
            print(f"  - {file}")
        print()


def main():
    # Define the folder path for demonstration
    folder_path = "file_organization_demo"

    # Step 1: Create sample files
    create_sample_files(folder_path)

    # Step 2: Organize files by their extensions
    organize_files_by_extension(folder_path)

    # Step 3: Display the organized folder structure
    display_folder_contents(folder_path)


if __name__ == "__main__":
    main()
```

```
Sample files created successfully.

Organizing files by extensions…

Moved: another_new_file.csv -> file_organization_demo\csv\another_new_file_1.csv
Moved: code1.py -> file_organization_demo\py\code1.py
Moved: code2.py -> file_organization_demo\py\code2.py
Moved: document1.pdf -> file_organization_demo\pdf\document1_1.pdf
Moved: image1.jpg -> file_organization_demo\jpg\image1_1.jpg
Moved: image2.png -> file_organization_demo\png\image2_1.png
Moved: new_file.txt -> file_organization_demo\txt\new_file.txt
Moved: notes1.txt -> file_organization_demo\txt\notes1_1.txt
Moved: presentation1.pptx -> file_organization_demo\pptx\presentation1_1.pptx
Moved: report1.pdf -> file_organization_demo\pdf\report1.pdf
Moved: report2.pdf -> file_organization_demo\pdf\report2.pdf
Moved: script1.py -> file_organization_demo\py\script1_1.py
Moved: spreadsheet1.xlsx -> file_organization_demo\xlsx\spreadsheet1_1.xlsx

Files have been organized successfully.

Folder structure after organization:

Folder: file_organization_demo

Folder: file_organization_demo\csv
  - another_new_file.csv
```

```
  - another_new_file_1.csv

Folder: file_organization_demo\jpg
  - image1.jpg
  - image1_1.jpg

Folder: file_organization_demo\pdf
  - document1.pdf
  - document1_1.pdf
  - report1.pdf
  - report2.pdf

Folder: file_organization_demo\png
  - image2.png
  - image2_1.png

Folder: file_organization_demo\pptx
  - presentation1.pptx
  - presentation1_1.pptx

Folder: file_organization_demo\py
  - code1.py
  - code2.py
  - script1.py
  - script1_1.py

Folder: file_organization_demo\txt
  - new_file.txt
  - notes1.txt
  - notes1_1.txt

Folder: file_organization_demo\xlsx
  - spreadsheet1.xlsx
  - spreadsheet1_1.xlsx
```

3 *Data Cleaning*

[3]:
```python
# Identify a repetitive task in your workflow and create python scripts a␣
↪autonomous it.This could include tasks like file norganisation, data␣
↪cleaning,or system maintenance...

import pandas as pd
import os


def clean_data(input_file,output_file):
```

```
    data = pd.read_csv(input_file)
    cleaned_data = data.dropna()
    cleaned_data.to_csv(output_file,index = False)
    print(f"Cleaned Data save to {output_file}")

if __name__ == "__main__":
    input_csv = input("Enter the path of csv file:")
    output_csv = input("Enter your updte file to exact path name what you want␣
 ↪to here:")
    clean_data(input_csv,output_csv)
```

```
Enter the path of csv file: Career.csv
Enter your updte file to exact path name what you want to here: Career_life.csv

Cleaned Data save to Career_life.csv
```

[13]:
```
import os

os.listdir(".")
```

[13]:
```
['.ipynb_checkpoints',
 '.~Csd_Data2.ipynb',
 '02 Churn-Dataset.xlsx',
 '02_Churn_Dataset.xlsx',
 '03 Diversity-Inclusion-Dataset.xlsx',
 'Advantages.csv',
 'Advantages_Data_File1.ipynb',
 'Advantages_plotly.plot2.ipynb',
 'Advantages_Plot_File1.ipynb',
 'Android_Csv_Data1.csv',
 'basha_bakery_order.csv',
 'Battery_life_comparison.csv',
 'Blasting_mobiles.csv',
 'Call-Center-Dataset.xlsx',
 'Call_Dataset_Center.csv',
 'Camera_quality.csv',
 'Career.csv',
 'Career_life.csv',
 'car_crashes.csv',
 'catboost_info',
 'Churn-Dataset.csv',
 'churn_dataset.xlsx',
 'codealpha2.ipynb',
 'codealpha2.pdf',
 'codealpha3.ipynb',
 'codealpha3.pdf',
 'codealpha4.ipynb',
 'Cproj_2-visu.ipynb',
```

```
'Csd3.ipynb',
'CSP_Kaif.docx',
'csp_proj-Copy1.ipynb',
'CSP_Proj.csv',
'customers1.csv',
'customers10.csv',
'customers2.csv',
'customers20.csv',
'customers3.csv',
'customers4.csv',
'customers5.csv',
'Customer_Ratings.csv',
'data.csv',
'data1.ipynb',
'data2.ipynb',
'Dataset.csv',
'Dataset1.csv',
'Dataset2.csv',
'Data_Adv.csv',
'Disadvantages.csv',
'Disadvantages_Plotly-Plot2.ipynb',
'Disadvatages_Data_File2.ipynb',
'Disadvatages_Plot_File2.ipynb',
'Diversity-Inclusion-Dataset.xlsx',
'diwali_sale.csv',
'Ecom_flp.csv',
'election.csv',
'EMP.csv',
'EMP1.csv',
'EMP2.csv',
'EMP3.csv',
'EMP_Clean.csv',
'excel_1.xlsx.xlsx',
'Features_popularity.csv',
'file.xlsx',
'file_organization_demo',
'flights.csv',
'flipkart.csv',
'Geographical_Ratings.csv',
'hotel_bookings.csv',
'House_price.csv',
'Indian_Mobile_Incidents_Data.csv',
'internship_with_codealpha.ipynb',
'internship_with_codealpha.pdf',
'iphone.csv',
'ipl.csv',
'ipl_matches.csv',
```

```
'jupyter1.ipynb',
'jupyter2.ipynb',
'jupyter3.ipynb',
'jupyter_pandas.ipynb',
'kaif1.pbix',
'kaifproject',
'Market_Share_by_Manufacture.csv',
'Merge_Datasets.csv',
'mobile_incdnt_dataset.csv',
'movies_list.pkl',
'mpg.csv',
'netflix_data.csv',
'ntb.ipynb',
'organized_files',
'OS_system.csv',
'pandas_day1.ipynb',
'path',
'practise.ipynb',
'PYTHON PROGRAMMING TASK.pdf',
'python_project_1.ipynb',
'Sales_Trends.csv',
'Screen_size.csv',
'similarity.pkl',
'smartphone.csv',
'Statistics.ipynb',
'Stats_Practice.ipynb',
'stu.csv',
'Student_file.csv',
'student_scores.csv',
'Stu_data.csv',
'T20.csv',
'Tata Project.ipynb',
'Test.csv',
'Titanic.csv',
'Titanic_Updated_Names.csv',
'Train.csv',
'Untitled video - Made with Clipchamp (1).mp4',
'Untitled video - Made with Clipchamp.mp4',
'Untitled.ipynb',
'Untitled1.ipynb',
'Untitled2.ipynb',
'Untitled3.ipynb',
'Untitled4.ipynb',
'Weather.csv',
'yash.png',
'yash1.png']
```

4 *System Maintenance*

[20]:
```
import os
import time

def create_sample_files(folder_path):
    """Create sample files and set their modification times."""
    os.makedirs(folder_path, exist_ok=True)
    sample_files = ["old_file1.txt", "old_file2.log", "new_file1.docx",␣
 ↪"new_file2.csv"]

    # Create files with content
    for file_name in sample_files:
        file_path = os.path.join(folder_path, file_name)
        with open(file_path, "w") as file:
            file.write(f"Sample content for {file_name}")

    # Modify the timestamps of files
    cutoff_time = time.time() - (30 * 86400)  # 30 days ago
    for file_name in ["old_file1.txt", "old_file2.log"]:
        file_path = os.path.join(folder_path, file_name)
        os.utime(file_path, (cutoff_time - 100, cutoff_time - 100))  # Simulate␣
 ↪older files

    print("Sample files created with simulated timestamps.\n")

def delete_old_files(folder_path, days):
    """Delete files older than a certain number of days."""
    cutoff_time = time.time() - (days * 86400)
    print(f"Deleting files older than {days} days...\n")

    for file_name in os.listdir(folder_path):
        file_path = os.path.join(folder_path, file_name)
        if os.path.isfile(file_path) and os.path.getmtime(file_path) <␣
 ↪cutoff_time:
            os.remove(file_path)
            print(f"Deleted: {file_name}")
    print("\nCleanup complete!")

def main():
    # Define the folder path
    folder_path = "file_cleanup_demo"

    # Step 1: Create sample files
    create_sample_files(folder_path)

    # Step 2: Delete files older than 30 days
```

```
    delete_old_files(folder_path, days=30)

    # Display the remaining files
    print("\nRemaining files in the folder:")
    for root, dirs, files in os.walk(folder_path):
        print(f"\n{root}")
        for file in files:
            print(f"  - {file}")

if __name__ == "__main__":
    main()
```

```
Sample files created with simulated timestamps.

Deleting files older than 30 days…

Deleted: old_file1.txt
Deleted: old_file2.log

Cleanup complete!

Remaining files in the folder:

file_cleanup_demo
  - new_file1.docx
  - new_file2.csv
```