**Assignment**

**CSA0805 – Python Programming**

|  |  |
| --- | --- |
| **Register Number** | **192372167** |
| **Name** | **BHARATH .P** |

1. **Title:**

**“Batch File Renamer: Add Prefixes, Suffixes, and Modify Extensions in Bulk”**

**Problem Statement:**

**In a busy office environment, you often need to organize and rename multiple files according to specific naming conventions. For instance, you might need to:**

1. **Add a consistent prefix or suffix to file names for easier categorization.**
2. **Change the file extension to standardize file formats.**
3. **Replace certain characters or strings in filenames to adhere to a naming policy.**

**Currently, this process is done manually, which is time-consuming and error-prone, especially when dealing with a large number of files.**

**Code:**

**import os**

**def rename\_files(directory, prefix='', suffix='', new\_extension=None, replace\_map=None):**

**"""**

**Renames files in the specified directory based on given criteria.**

**:param directory: The directory containing files to rename.**

**:param prefix: Optional prefix to add to file names.**

**:param suffix: Optional suffix to add to file names.**

**:param new\_extension: Optional new extension for the files (e.g., '.txt').**

**:param replace\_map: Dictionary where keys are characters/strings to replace and values are replacements.**

**"""**

**if not os.path.isdir(directory):**

**raise NotADirectoryError(f"The directory {directory} does not exist.")**

**# Get a list of files in the directory**

**for filename in os.listdir(directory):**

**# Construct the full file path**

**old\_file\_path = os.path.join(directory, filename)**

**# Skip directories**

**if os.path.isdir(old\_file\_path):**

**continue**

**# Apply the prefix and suffix**

**new\_filename = f"{prefix}{filename}{suffix}"**

**# Apply replacements**

**if replace\_map:**

**for old, new in replace\_map.items():**

**new\_filename = new\_filename.replace(old, new)**

**# Change the file extension if needed**

**if new\_extension:**

**base, \_ = os.path.splitext(new\_filename)**

**new\_filename = f"{base}{new\_extension}"**

**# Construct the new file path**

**new\_file\_path = os.path.join(directory, new\_filename)**

**# Rename the file**

**os.rename(old\_file\_path, new\_file\_path)**

**print(f"Renamed '{filename}' to '{new\_filename}'")**

**# Example usage**

**if \_\_name\_\_ == "\_\_main\_\_":**

**# Set the directory and renaming parameters**

**directory = './your\_directory'**

**prefix = 'new\_'**

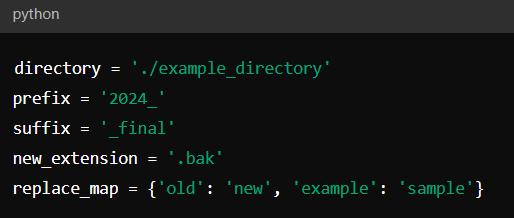
**suffix = '\_2024'**

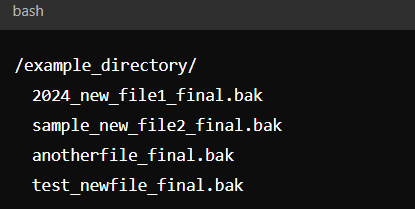
**new\_extension = '.txt'**

**replace\_map = {'old': 'new', 'example': 'sample'}**

**rename\_files(directory, prefix, suffix, new\_extension, replace\_map)**

**Output Screen Shots:**

****

****

**Conclusion:**

**The Python script effectively addresses the need for automated file renaming by providing a flexible and efficient solution. By allowing users to add prefixes and suffixes, change file extensions, and replace substrings, the script simplifies the process of organizing and standardizing file names. This automation reduces manual effort, minimizes errors, and ensures consistent file naming conventions, ultimately saving time and enhancing productivity in managing large volumes of files.**