Faculty of Engineering and Technology Electrical and Computer Engineering Department

ENC3130 – LINUX LABORATORY
Project_#2

Prepared by: Diana Naseer
Student ID: 1210363
Hala Jebreel
1210606.

Instructor: Aziz Qaroush.

Section: 2.
Date: 12/6/2024.

Theory:

- Imports needed in the project:

```python
# linux.py
#=====================================================================
import argparse
import json
import logging
import os
from commands import mv_last, categorize, count_files, delete_file, rename_file, list_files, sort_files
#---------------------------------------------------------------------
```

- Load configuration file from json file:

```python
# Function to load the configuration file
1 usage
def load_configuration(configuration_path):
    with open(configuration_path, 'r', encoding='utf-8') as x:
        configuration = json.load(x)
    return configuration
#=====================================================================
```

- Function to excecute script :

```python
# Function to execute the script
1 usage
def execute_script(The_script_path, configuration, The_log_output):
    The_commands = {
        'mv_last': mv_last,
        'categorize': categorize,
        'count_files': count_files,
        'delete_file': delete_file,
        'rename_file': rename_file,
        'list_files': list_files,
        'sort_files': sort_files
    }
```

```
    with open(The_script_path, 'r', encoding='utf-8') as x:
        ScriptLines = x.readlines()

    results = []
    for line in ScriptLines:
        Splited_parts = line.strip().split()
        CMD_Name = Splited_parts[0]
        args = Splited_parts[1:]

        if CMD_Name in The_commands:
            command = The_commands[CMD_Name]
            if CMD_Name == 'categorize':
                if len(args) == 1:
                    args.append(configuration['Threshold_size'])
                elif len(args) == 2:
                    pass
                else:
                    results.append((line.strip(), 'FAILURE'))
                    continue
            result = command(*args)
            results.append((line.strip(), 'SUCCESS' if result is None else result))
        else:
            results.append((line.strip(), 'FAILURE'))

    output_results(results, The_log_output, configuration['Output'], configuration['Same_dir'])
#=============================================================================
```

- Function output result:

```
1 usage
def output_results(The_results, TheLog_output, OutputFormat, same_dir):
    if not TheLog_output:
        raise ValueError("Output log file the  path is empty!!")
    # ============================================================================
    log_diriction = os.path.dirname(TheLog_output)
    if log_diriction and not os.path.exists(log_diriction):
        os.makedirs(log_diriction) # Create the directory if it doesn't exist
    # ============================================================================
    if OutputFormat == 'csv':
        import csv
        csv_path = TheLog_output.replace('.log', '.csv')
        with open(csv_path, 'w', newline='', encoding='utf-8') as csvfile:
            writer = csv.writer(csvfile)
            writer.writerow(['Command', 'Result'])
            for result in The_results:
                writer.writerow(result)
        logging.info(f"Results written to {csv_path}")
    else:
        with open(TheLog_output, 'w', encoding='utf-8') as f:
            for result in The_results:
                f.write(f"{result[0]}: {result[1]}\n")
        logging.info(f"Results written to {TheLog_output}")
```

- Function setup logging:

```python
# Function to setup the logging
1 usage
def setup_logging(log_dir, max_log_files):
    if not log_dir:
        log_dir = "logs"
    if not os.path.exists(log_dir):
        os.makedirs(log_dir)

    # ========================================================================
    TheLogFiles = [os.path.join(log_dir, x) for x in os.listdir(log_dir) if x.endswith('.log')]
    TheLogFiles.sort(key=os.path.getmtime)

    while len(TheLogFiles) >= max_log_files:
        TheOldestLog = TheLogFiles.pop(0)
        os.remove(TheOldestLog)
        logging.info(f"Removed old log file: {TheOldestLog}")

    logging.basicConfig(filename=os.path.join(log_dir, 'execution.log'), level=logging.INFO,
                        format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', encoding='utf-8')
```

- Main:

```python
1 usage
def main():
    parser = argparse.ArgumentParser(description='Script Executor')
    # ========================================================================
    parser.add_argument( *name_or_flags: '-i', '--input', required=True, help='Input script file')
    # ========================================================================
    parser.add_argument( *name_or_flags: '-o', '--output', required=True, help='Output log file')
    # ========================================================================
    parser.add_argument( *name_or_flags: '-c', '--config', default='my_file.json', help='Configuration file')
    # ========================================================================
    args = parser.parse_args()
    # ========================================================================
    config = load_configuration(args.config)
    setup_logging(os.path.dirname(args.output), config['Max_log_files'])

    execute_script(args.input, config, args.output)
#============================================================================
if __name__ == '__main__':
    main()
```

- **2nd code for command**:

```python
2 usages
def my_last(src_directory, des_directory):
    try:
        The_logger.info(f"Source Directory: {src_directory}")
        The_logger.info(f"Destination Directory: {des_directory}")
        # ========================================================================
        # Check if source directory exists
        if not os.path.exists(src_directory):
            TheErrorMessage = f"Source directory {src_directory} does not exist."
            The_logger.error(TheErrorMessage)
            return TheErrorMessage
        # ========================================================================
        # Check if destination directory exists
        if not os.path.exists(des_directory):
            The_logger.info(f"Destination directory {des_directory} does not exist. Creating directory.")
            os.makedirs(des_directory)
        # ========================================================================
        # Check if source directory is a directory
        if not os.path.isdir(src_directory):
            TheErrorMessage = f"Source directory {src_directory} is not a directory."
            The_logger.error(TheErrorMessage)
            return TheErrorMessage
        # ========================================================================
        # Check if destination directory is a directory
        if not os.path.isdir(des_directory):
            # ========================================================================
            # If destination directory is not a directory, create it
            os.makedirs(des_directory)
            TheErrorMessage = f"Destination directory {des_directory} is not a directory."
```

```
            return TheErrorMessage
        # ========================================================================
        # Check if source directory is empty
        if not os.listdir(src_directory):
            TheErrorMessage = f"Source directory {src_directory} is empty."
            The_logger.error(TheErrorMessage)
            return TheErrorMessage
        # ========================================================================
        # Get the list of files in the source directory
        The_files = [os.path.join(src_directory, f) for f in os.listdir(src_directory)]

        The_files = [f for f in The_files if os.path.isfile(f)]
        # ========================================================================
        # Check if there are any files in the source directory!
        if not The_files:
            warning_message = f"No The_files to move in {src_directory}"
            The_logger.warning(warning_message)
            return warning_message
# ============================================================================
        # Move the latest file to the destination directory
        ThelatestFile = max(The_files, key=os.path.getmtime)
        The_logger.info(f"Latest File to Move: {ThelatestFile}")
        # ========================================================================
        # Move the latest file to the destination directory
        shutil.move(ThelatestFile, des_directory)
        success_message = f"Moved {ThelatestFile} to {des_directory}"
        The_logger.info(success_message)
        return None
    # ============================================================================
```

- **Category command:**

```
def categorize(directory, ThresholdSize):
    try:
        SmallDirication = os.path.join(directory, 'small_files')
        largeDiriction = os.path.join(directory, 'large_files')
        os.makedirs(SmallDirication, exist_ok=True)
        os.makedirs(largeDiriction, exist_ok=True)
        # ========================================================================
        ThresholdSize = parse_size(ThresholdSize) # Convert threshold size to bytes

        for filename in os.listdir(directory):# Loop through all files in the directory
            filepath = os.path.join(directory, filename)
            if os.path.isfile(filepath):# Check if it's a file
                if os.path.getsize(filepath) < ThresholdSize:
                    destination = os.path.join(SmallDirication, filename)
                else:# Move the file to the large_files directory
                    destination = os.path.join(largeDiriction, filename)

                if os.path.exists(destination):# Check if the destination file already exists
                    base, extension = os.path.splitext(destination)
                    counter = 1
                    NewDestination = f"{base}_{counter}{extension}"
                    while os.path.exists(NewDestination):
                        counter += 1
                        NewDestination = f"{base}_{counter}{extension}"
                    destination = NewDestination

                shutil.move(filepath, destination)# Move the file to the destination
        The_logger.info(f"Categorized files in {directory} by size {ThresholdSize} bytes")
```

- **Count file command:**

```
def count_files(directory):# Count the number of files in a directory
    try:# Try to count the number of files
        count = len([f for f in os.listdir(directory) if os.path.isfile(os.path.join(directory, f))])
        The_logger.info(f"Counted {count} files in {directory}")
        return count
    # ========================================================================
    except Exception as Exp:
        error_message = f"Error counting files: {Exp}"
        The_logger.error(error_message)
        return error_message
```

- **Delet file command :**

```
2 usages
def delete_file(filename, directory):
    try:
        filepath = os.path.join(directory, filename)
        if os.path.exists(filepath):
            os.remove(filepath)
            The_logger.info(f"Deleted {filename} from {directory}")
            return None
        # ========================================================================
        else:
            warning_message = f"File {filename} not found in {directory}"
            The_logger.warning(warning_message)
            return warning_message
    # ========================================================================
    except Exception as Exp:# Catch any errors
        error_message = f"Error deleting file: {Exp}"
        The_logger.error(error_message)
        return error_message
```

- **Rename file command:**

```
2 usages
def rename_file(TheoldName, TheNewname, Directory):
    try:
        OlDfilepath = os.path.join(Directory, TheoldName)
        # ========================================================================
        NewFilepath = os.path.join(Directory, TheNewname)
        # ========================================================================
        if os.path.exists(OlDfilepath):# Check if the file exists
            os.rename(OlDfilepath, NewFilepath)
            The_logger.info(f"Renamed {TheoldName} to {TheNewname} in {Directory}")
            return None
        # ========================================================================
        else:# If the file doesn't exist
            warning_message = f"File {TheoldName} not found in {Directory}"
            The_logger.warning(warning_message)
            return warning_message
        # ========================================================================
    except Exception as EXP:
        error_message = f"Error renaming file: {EXP}"
        The_logger.error(error_message)
        return error_message
```

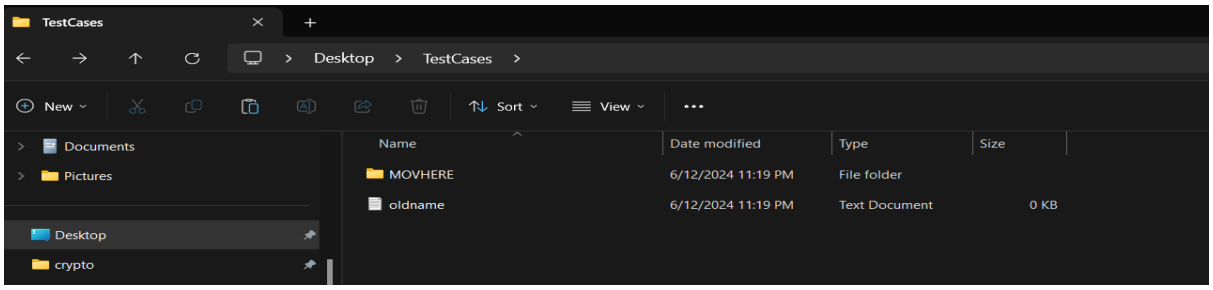
- **List file command:**

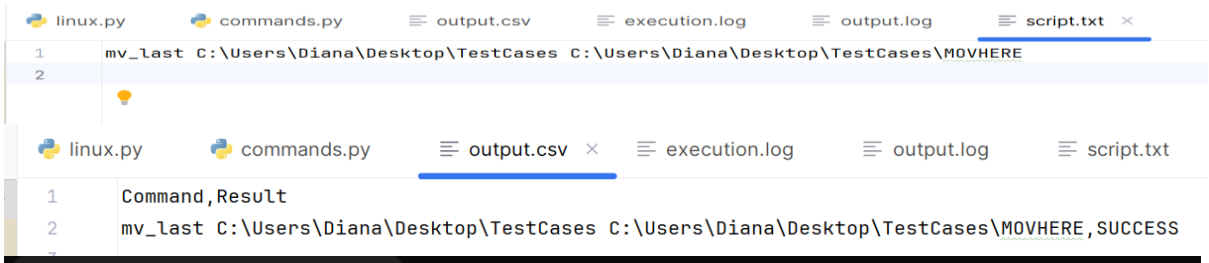
```
def list_files(directory):
    # ================================================================================
    try:# Try to list the files
        files = os.listdir(directory)
        The_logger.info(f"Files in {directory}: {files}")
        return files
    # ================================================================================
    except Exception as EXP:
        error_message = f"Error listing files: {EXP}"
        The_logger.error(error_message)
        return error_message
```

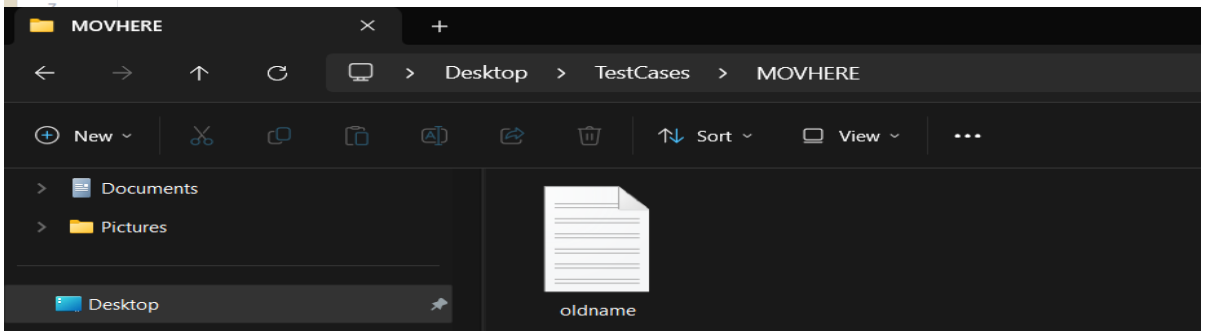
- **Sort command:**

```
def sort_files(directory, criteria):
    try:# Try to sort the files
        files = [os.path.join(directory, f) for f in os.listdir(directory) if os.path.isfile(os.path.join(directory, f))]
        # ========================================================================
        if criteria == 'name':
            files.sort(key=lambda x: os.path.basename(x))
        # ========================================================================
        elif criteria == 'date':
            files.sort(key=lambda x: os.path.getmtime(x))
        # ========================================================================
        elif criteria == 'size':
            files.sort(key=lambda x: os.path.getsize(x))
        # ========================================================================
        else:
            warning_message = f"Unsupported sorting criteria: {criteria}"
            The_logger.warning(warning_message)
        # ========================================================================
            return warning_message
        # ========================================================================
        SortedFiles = [os.path.basename(f) for f in files]
        The_logger.info(f"Sorted files in {directory} by {criteria}: {SortedFiles}")
        return SortedFiles
    # ============================================================================
    except Exception as e:# Catch any errors
        error_message = f"Error sorting files: {e}"
        The_logger.error(error_message)
        return error_message
```

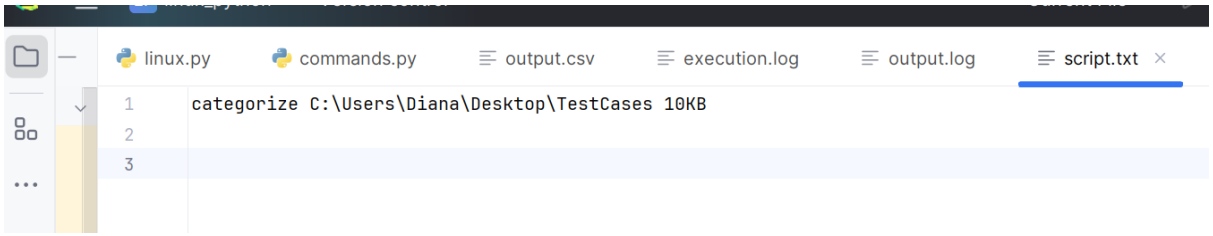
Test Cases:

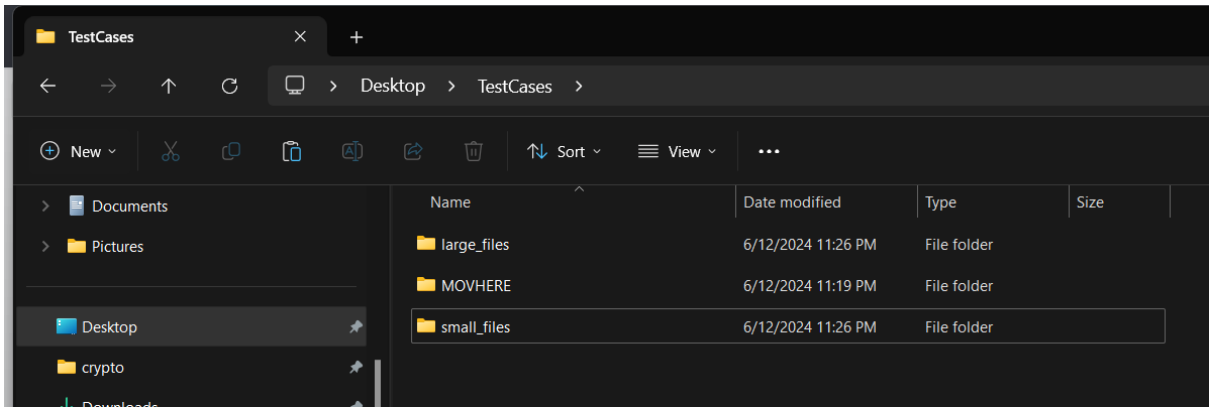

TestCases
Desktop
TestCases
New
Sort
View
Documents
Pictures
Desktop
crypto
Name
Date modified
Type
Size
MOVHERE
6/12/2024 11:19 PM
File folder
oldname
6/12/2024 11:19 PM
Text Document
0 KB

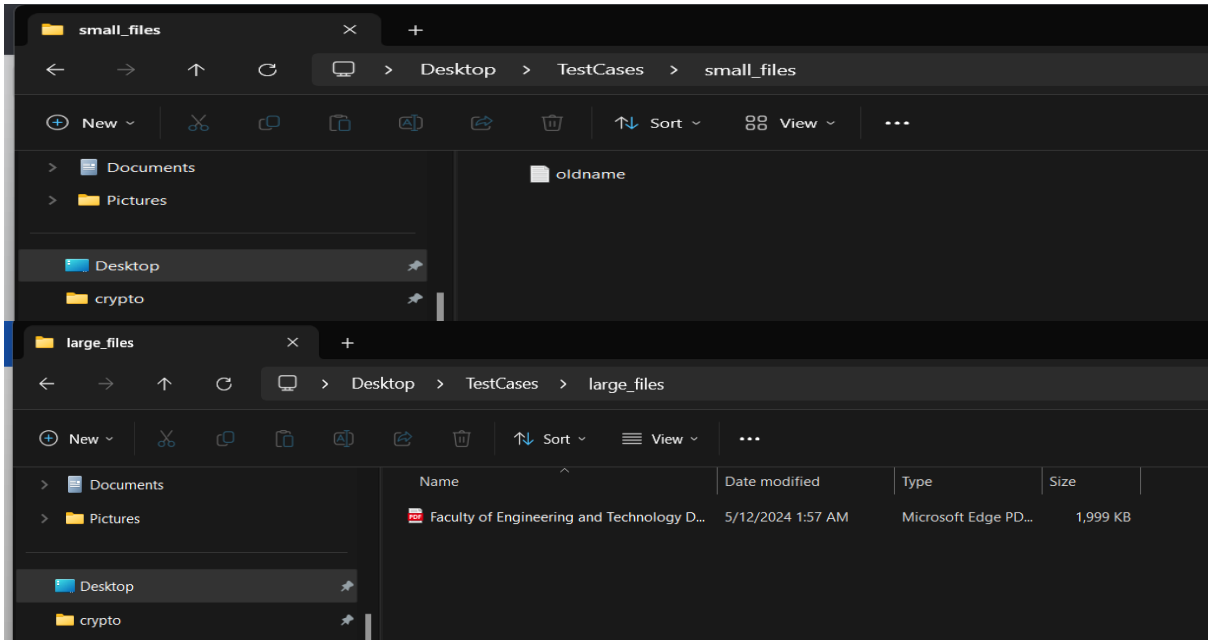

linux.py
commands.py
output.csv
execution.log
output.log
script.txt
1 mv_last C:\Users\Diana\Desktop\TestCases C:\Users\Diana\Desktop\TestCases\MOVHERE
2
linux.py
commands.py
output.csv
execution.log
output.log
script.txt
1 Command,Result
2 mv_last C:\Users\Diana\Desktop\TestCases C:\Users\Diana\Desktop\TestCases\MOVHERE,SUCCESS


MOVHERE
Desktop
TestCases
MOVHERE
New
Sort
View
Documents
Pictures
Desktop
oldname

==========

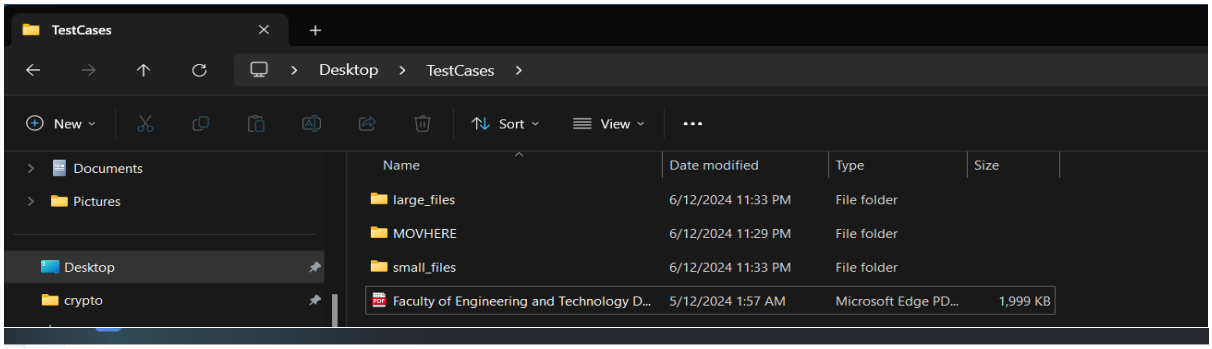

linux.py
commands.py
output.csv
execution.log
output.log
script.txt
1 categorize C:\Users\Diana\Desktop\TestCases 10KB
2
3

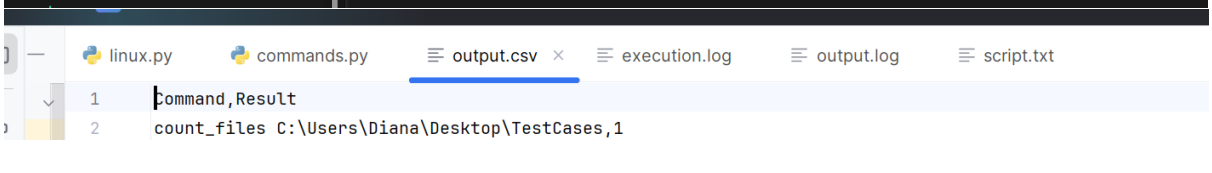

TestCases
Desktop
TestCases
New
Sort
View
Documents
Pictures
Desktop
crypto
Name
Date modified
Type
Size
large_files
6/12/2024 11:26 PM
File folder
MOVHERE
6/12/2024 11:19 PM
File folder
small_files
6/12/2024 11:26 PM
File folder

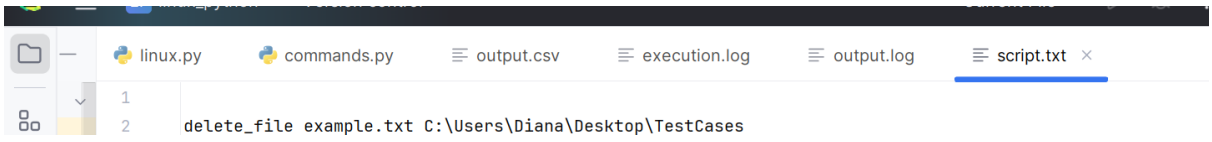


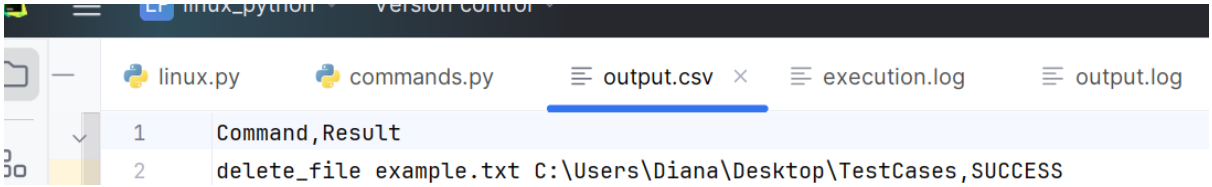
==========

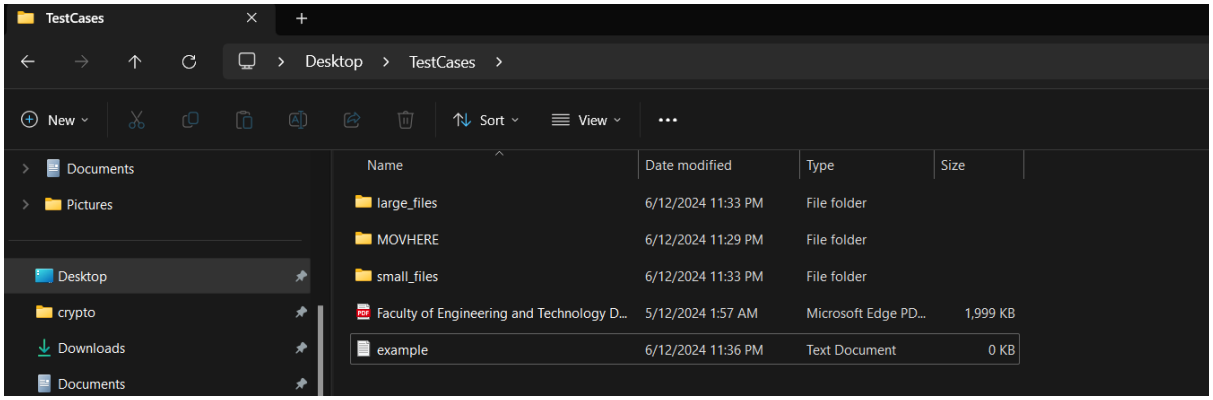
```
count_files C:\Users\Diana\Desktop\TestCases
```

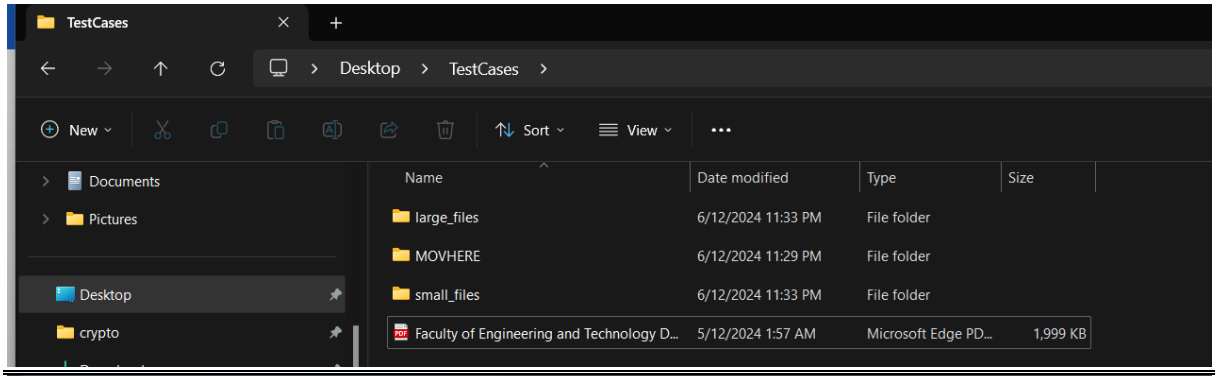


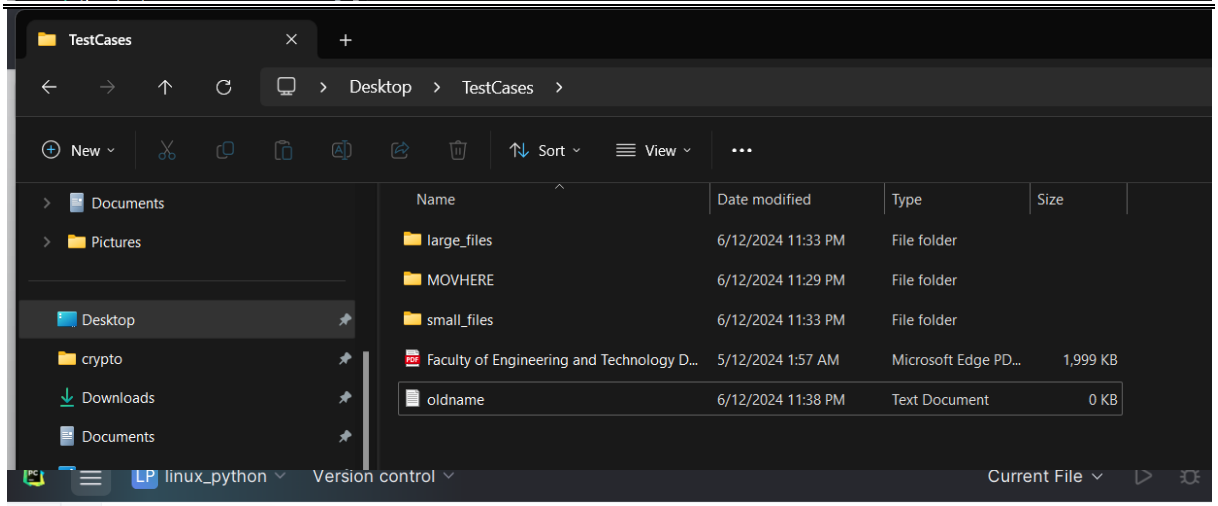


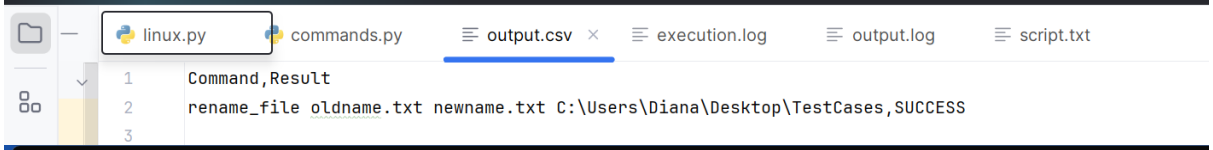


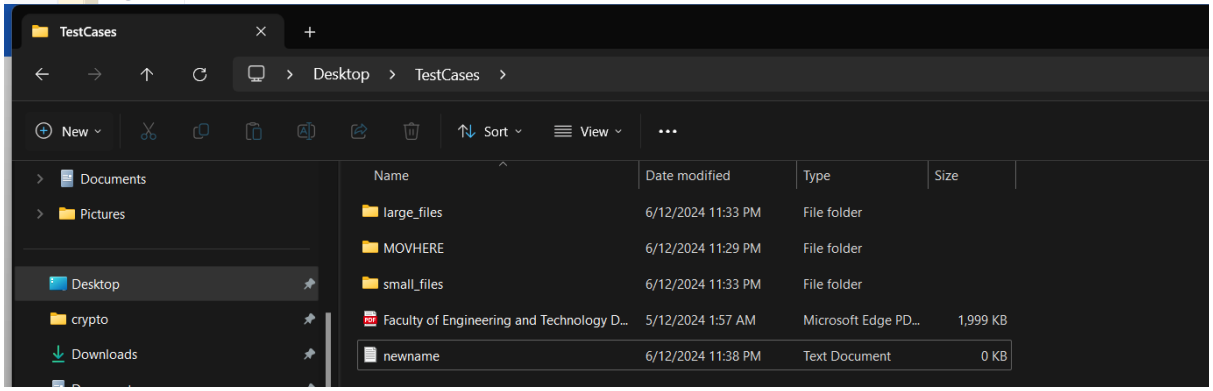


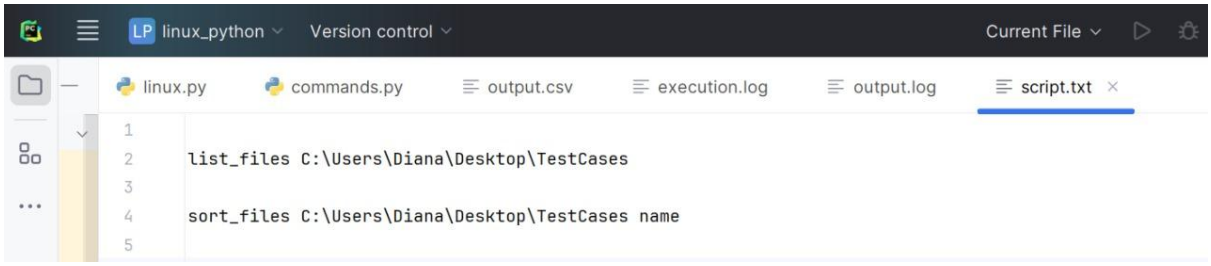

TestCases
Desktop
TestCases
New
Sort
View
Documents
Pictures
Desktop
crypto
Downloads
Documents
Name
Date modified
Type
Size
large_files
6/12/2024 11:33 PM
File folder
MOVHERE
6/12/2024 11:29 PM
File folder
small_files
6/12/2024 11:33 PM
File folder
Faculty of Engineering and Technology D...
5/12/2024 1:57 AM
Microsoft Edge PD...
1,999 KB
example
6/12/2024 11:36 PM
Text Document
0 KB

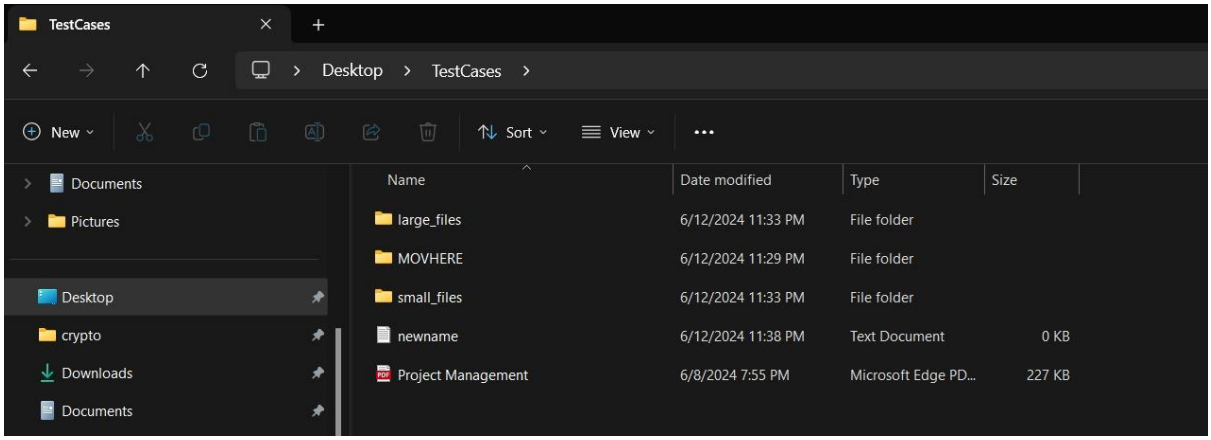

TestCases
Desktop
TestCases
New
Sort
View
Documents
Pictures
Desktop
crypto
Name
Date modified
Type
Size
large_files
6/12/2024 11:33 PM
File folder
MOVHERE
6/12/2024 11:29 PM
File folder
small_files
6/12/2024 11:33 PM
File folder
Faculty of Engineering and Technology D...
5/12/2024 1:57 AM
Microsoft Edge PD...
1,999 KB


TestCases
Desktop
TestCases
New
Sort
View
Documents
Pictures
Desktop
crypto
Downloads
Documents
Name
Date modified
Type
Size
large_files
6/12/2024 11:33 PM
File folder
MOVHERE
6/12/2024 11:29 PM
File folder
small_files
6/12/2024 11:33 PM
File folder
Faculty of Engineering and Technology D...
5/12/2024 1:57 AM
Microsoft Edge PD...
1,999 KB
oldname
6/12/2024 11:38 PM
Text Document
0 KB
linux_python
Version control
Current File


linux.py
commands.py
output.csv
execution.log
output.log
script.txt
1 Command,Result
2 rename_file oldname.txt newname.txt C:\Users\Diana\Desktop\TestCases,SUCCESS
3


TestCases
Desktop
TestCases
New
Sort
View
Documents
Pictures
Desktop
crypto
Downloads
Name
Date modified
Type
Size
large_files
6/12/2024 11:33 PM
File folder
MOVHERE
6/12/2024 11:29 PM
File folder
small_files
6/12/2024 11:33 PM
File folder
Faculty of Engineering and Technology D...
5/12/2024 1:57 AM
Microsoft Edge PD...
1,999 KB
newname
6/12/2024 11:38 PM
Text Document
0 KB


linux_python
Version control
Current File
linux.py
commands.py
output.csv
execution.log
output.log
script.txt
1
2 list_files C:\Users\Diana\Desktop\TestCases
3
4 sort_files C:\Users\Diana\Desktop\TestCases name
5


TestCases
Desktop
TestCases
New
Sort
View
Documents
Pictures
Desktop
crypto
Downloads
Documents
Name
Date modified
Type
Size
large_files 6/12/2024 11:33 PM File folder
MOVHERE 6/12/2024 11:29 PM File folder
small_files 6/12/2024 11:33 PM File folder
newname 6/12/2024 11:38 PM Text Document 0 KB
Project Management 6/8/2024 7:55 PM Microsoft Edge PD... 227 KB