

EXPERIMENT NO:-03

Name - Yash Wagh

Roll - 122A1125

Div - D

Batch-D3

AIM:- To Implement all file and directory functions.

THEORY:-

File and directory handling in Python is essential for reading, writing, and managing files and directories on the filesystem. Python provides a rich set of built-in modules and functions to facilitate these operations. Here are some general concepts and modules used for file and directory handling in Python:

File Handling:

Opening and Closing Files:

You can use the `open()` function to open a file in different modes like read mode ('r'), write mode ('w'), append mode ('a'), or binary mode ('b'). After performing operations on the file, it's essential to close the file using the `close()` method to free up system resources.

Reading from Files:

Use methods like `read()`, `readline()`, or `readlines()` to read content from the file.

`read()` reads the entire file, `readline()` reads a single line, and `readlines()` reads all lines into a list.

Writing to Files:

Use the `write()` method to write content to a file. It is essential to open the file in write or append mode ('w' or 'a') before writing to it.

Appending to Files:

Open a file in append mode ('a') to add content to the end of the file

without overwriting existing content.

2

Error Handling:

Use exception handling (try-except) to handle file-related errors like `FileNotFoundError`, `PermissionError`, or `IOError`.

Directory Handling:

Listing Directory Contents:

Use the `os.listdir()` function to get a list of all files and directories in a given directory.

This function returns a list of strings where each string represents the name of a file or directory in the specified directory.

Creating Directories: Use the `os.makedirs()` function to create directories recursively. This function creates all intermediate-level directories needed to create the specified directory.

Removing Directories:

Use the `os.rmdir()` function to remove directories. It raises an `OSError` if the directory is not empty. Alternatively, use `shutil.rmtree()` to remove a directory and all its contents recursively.

Navigating Directories:

Use functions like `os.getcwd()` to get the current working directory and `os.chdir()` to change the current working directory.

Filesystem Operations:

Python's `os` module provides various functions for filesystem operations such as renaming files (`os.rename()`), removing files (`os.remove()`), checking if a path exists (`os.path.exists()`), and more.

Error Handling:

Like file handling, directory handling also requires error handling to deal with potential issues such as permission errors, file not found errors, or disk space issues.

Python's file and directory handling capabilities are powerful and versatile, allowing developers to perform a wide range of operations on files and directories efficiently and effectively.

PROGRAM:-

```
file = open('Exp03.txt','w')
file.write("This is the write command\n")
file.write("It allows us to write in a particular file\n")
file.write('Welcome to SIES GST\n')
file = open('Exp03.txt', 'a')
file.write("This will add this line")
file = open('Exp03.txt', 'r')
#-----
for each in file:
    print (each)
file= open('Exp03.txt','r')
content=file.read()
print('content using read():',content)
file= open('Exp03.txt','r')
content=file.readline()
print('content using readline():',content)
file= open('Exp03.txt','r')
content=file.readlines()
print('content using readlines():',content)
file.close()
#-----
file_path = "Exp04.txt"
try:
    # Attempt to open the file in read mode
    with open(file_path, "r") as file:
        # Read content from the file
        content = file.read()
        print("Content of the file:")
        print(content)
except FileNotFoundError:
    # Handle the case where the file does not exist
    print(f"Error: File '{file_path}' not found.")
except PermissionError:
    # Handle the case where permission to open the file is denied
    print(f"Error: Permission denied to open the file '{file_path}'.")
except Exception as e:
    # Handle any other exceptions that might occur
    print(f"An error occurred: {e}")
#-----
import os
from pathlib import Path
```

```
def create_directory(directory_name):
    try:
        os.mkdir(directory_name)
        print(f"Directory '{directory_name}' created successfully.")
    except FileExistsError:
        print(f"Directory '{directory_name}' already exists.")
#-----
import os

def delete_directory(directory_name):
    try:
        os.rmdir(directory_name)
        print(f"Directory '{directory_name}' deleted successfully.")
    except FileNotFoundError:
        print(f"Directory '{directory_name}' does not exist.")
    except OSError as e:
        print(f"Error deleting directory '{directory_name}': {e}")
#-----

def check_directory_exists(directory_name):
    if os.path.exists(directory_name):
        print(f"Directory '{directory_name}' exists.")
    else:
        print(f"Directory '{directory_name}' does not exist.")
#-----
import os

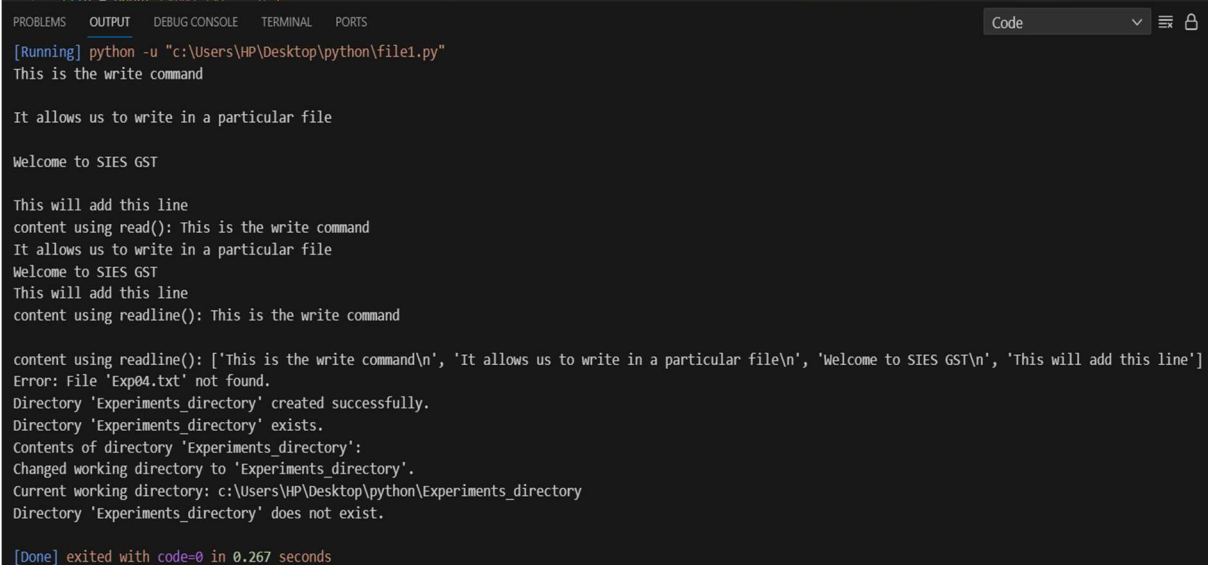
def list_directory_contents(directory_name):
    try:
        contents = os.listdir(directory_name)
        print(f"Contents of directory '{directory_name}':")
        for item in contents:
            print(item)
    except FileNotFoundError:
        print(f"Directory '{directory_name}' does not exist.")
#-----

def change_working_directory(new_directory):
    try:
        os.chdir(new_directory)
        print(f"Changed working directory to '{new_directory}'.")
    except FileNotFoundError:
        print(f"Directory '{new_directory}' does not exist.")
    except PermissionError:
        print(f"No permission to access directory '{new_directory}'.")
#-----

def get_current_working_directory():
```

```
cwd = os.getcwd()
print(f"Current working directory: {cwd}")
if __name__ == "__main__":
    # Test the functions
    create_directory("Experiments_directory")
    check_directory_exists("Experiments_directory")
    list_directory_contents("Experiments_directory")
    change_working_directory("Experiments_directory")
    get_current_working_directory()
    delete_directory("Experiments_directory")
#-----
```

OUTPUT:-



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code
[Running] python -u "c:\Users\HP\Desktop\python\file1.py"
This is the write command

It allows us to write in a particular file

Welcome to SIES GST

This will add this line
content using read(): This is the write command
It allows us to write in a particular file
Welcome to SIES GST
This will add this line
content using readline(): This is the write command

content using readline(): ['This is the write command\n', 'It allows us to write in a particular file\n', 'Welcome to SIES GST\n', 'This will add this line']
Error: File 'Exp04.txt' not found.
Directory 'Experiments_directory' created successfully.
Directory 'Experiments_directory' exists.
Contents of directory 'Experiments_directory':
Changed working directory to 'Experiments_directory'.
Current working directory: c:\Users\HP\Desktop\python\Experiments_directory
Directory 'Experiments_directory' does not exist.

[Done] exited with code=0 in 0.267 seconds
```

CONCLUSION:-Hence we were able to implement all file and directory functions.