Experiment No. 5
Exploring Files and directories: Python program to append data to existing file and then display the entire file
Date of Performance:
Date of Submission:

Experiment No. 5

Title: Exploring Files and directories: Python program to append data to existing file and then display the entire file

Aim: To Exploring Files and directories: Python program to append data to existing file and then display the entire file

Objective: To Exploring Files and directories

Theory:

Directory also sometimes known as a folder are unit organizational structure in computer's file system for storing and locating files or more folders. Python now supports a number of APIs to list the directory contents. For instance, we can use the Path.iterdir, os.scandir, os.walk, Path.rglob, or os.listdir functions.

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but alike other concepts of Python, this concept here is also easy and short. Python treats file differently as

text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with Reading and Writing files.

Working of open() function

We use open () function in Python to open a file in read or write mode. As explained above, open ( ) will return a file object. To return a file object we use open() function along with two arguments, that accepts file name and the mode, whether to read or write. So, the syntax being: open(filename, mode). There are three kinds of mode, that Python provides and how files can be opened:

" r ", for reading.

" w ", for writing.

" a ", for appending.

" r+ ", for both reading and writing

Code:

```
# Open the file in write mode and write initial data
file1 = open("myfile.txt", "w")
L = ["This is Delhi \n", "This is Paris \n", "This is London"]
file1.writelines(L)
file1.close()


# Append additional data to the file
file1 = open("myfile.txt", "a")  # Open the file in append mode
file1.write("Today \n")
file1.close()


# Read and print the contents of the file after appending
file1 = open("myfile.txt", "r")
print("Output of Readlines after appending")
print(file1.read())
print()
file1.close()
```

```
# Open the file in write mode and overwrite existing data
file1 = open("myfile.txt", "w")
file1.write("Tomorrow \n")
file1.close()

# Read and print the contents of the file after overwriting
file1 = open("myfile.txt", "r")
print("Output of Readlines after writing")
print(file1.read())
print()
file1.close()
```

Output:

```
Microsoft Windows [Version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\>python fileappend.py
Output of Readlines after appending
This is Delhi
This is Paris
This is LondonToday


Output of Readlines after writing
Tomorrow



D:\>
```

Conclusion: Directories and files have been explored.

The Python program efficiently demonstrates appending data to an existing file and then displaying the entire file content. It showcases file manipulation techniques by appending data in append mode, reading and printing the complete file contents, and overwriting data using write mode, providing a concise exploration of file handling in Python.