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.

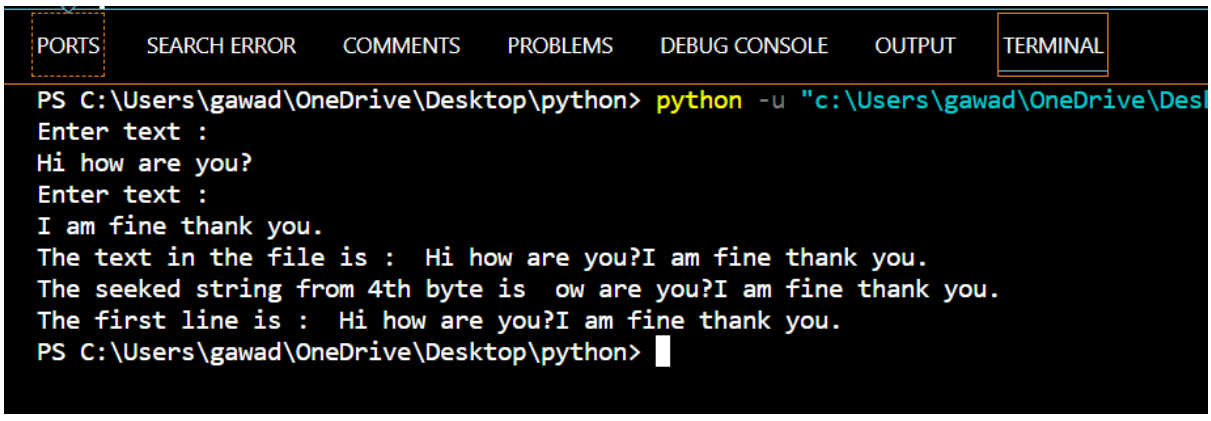
" r+ ", for both reading and writing

Code:

```
# Write function
f = open('pmg.txt', 'w')
str = input("Enter text : \n")
f.write(str)
f.close()

# Append Function
f = open('pmg.txt', 'a')
str = input("Enter text : \n")
f.write(str)
f.close()

# Read Function
f = open('pmg.txt', 'r')
str1 = f.read()
print("The text in the file is : ", str1)
f.close()
```

```
# Accessign string from a specific position

f = open('pmg.txt', 'r')

f.seek(4)

print("The seeked string from 4th byte is ", f.read())

f.close()


#ReadLine

f = open('pmg.txt', 'r')

print("The first line is : ", f.readline())

f.close()
```

Output:



Conclusion:

This Python code snippet demonstrates basic file handling operations such as writing to a file, appending to a file, reading from a file, seeking a specific position in a file, and reading a single line from a file. It begins by prompting the user to input text, which is then written to a file named "pmg.txt" using the write mode ('w'). Subsequently, additional text inputted by the user is appended to the same file using the append mode ('a'). The content of the file is then read and displayed to the user. Additionally, the code seeks and prints the string from the 4th byte position and reads the first line of the file.

Overall, this code demonstrates the fundamental operations involved in file handling in Python, providing a clear illustration of how to write, append, read, seek, and read specific lines from a file. It highlights the simplicity and versatility of Python's file handling capabilities, making it easy to work with external files for various purposes in programming projects.