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

Program To Take Multiline Input

CODE:

```
f=open("xyz.txt","w")
while True:

    user_input = input("Enter a string (type '@' to stop): ")

    if user_input == '@':
        break

    f.write(user_input + '\n')

f.close()
f=open("xyz.txt","r")
p=f.read()
print(p)

f.close()
```

OUTPUT:

```
============= RESTART: C:\Vedanti_Degree\SEM_4\SBL_PYTHON\pr5.2.py =============
Enter a string (type '@' to stop): VR
Enter a string (type '@' to stop): DP
Enter a string (type '@' to stop): @
VR
DP
```

Program To Append Data To Existing File And Then Display The Entire File

CODE:

```
f=open("xyz.txt","a+")
str=input("Enter a string : ")

f.write(str)
f.seek(0,0)


l=f.read()
print(l)

f.close()
```

OUTPUT:

```
============= RESTART: C:\Vedanti_Degree\SEM_4\SBL_PYTHON\pr5.3.py =============
Enter a string : Hello Pyton
VR
DP
Hello Pyton
```

Program to Create Directory

```
import os

def create_directory(directory_path):
    try:
        os.makedirs(directory_path)
        print(f"Directory '{directory_path}' created successfully.")
    except FileExistsError:
        print(f"Directory '{directory_path}' already exists.")

# Example usage:
directory_path = "my_directory"
create_directory(directory_path)
```

OUTPUT:

```
============= RESTART: C:/Vedanti_Degree/SEM_4/SBL_PYTHON/pr5.4.py =============
Directory 'my_directory' created successfully.
```

CONCLUSION:

In conclusion, the exploration of files and directories in Python has provided a comprehensive understanding of how to interact with the filesystem effectively. Through practical exercises and demonstrations, we have learned essential techniques for reading from and writing to files, navigating directories, and manipulating file paths.