NAME: Meet Raut

DIV: S2-1

ROLL NO: 2201084

EXPERIMENT – 4:

- ***AIM:*** **To study and implement programs on exploring Files and directories:**
  - **a) To append data to existing file and then display the entire file.**
  - **b) To count number of lines, words and characters in a file. .**
  - **c) Python program To display file available in current directory.**

- ***THEORY:***

a) Append Data to Existing File and Display the Entire File:

In Python, you can use the built-in open() function to open a file in different modes, such as 'r' for reading, 'w' for writing, and 'a' for appending. To append data to an existing file, you can open the file in append mode and then use the write() method to add new content. After appending, you can read and display the entire file.

- **PROGRAM 1:**

```
file=open('hello.txt','r')
print(file.read())

file=open('hello.txt','w')
file.write("Writing PYTHON LAB\n")

file=open('hello.txt','a')
file.write("Appending an text TSEC\n")
file.close()
```

- **OUTPUT:**

```
Writing PYTHON LAB
Appending an text TSEC
This is extra test IndiaThis is extra test India
>>>
```

- **PROGRAM 2:**

```
file1 = open("myfile.txt", "w")
L = ["This is Delhi \n", "This is Paris \n", "This is London"]
file1.writelines(L)
file1.close()

# Append-adds at last
# append mode
file1 = open("myfile.txt", "a")

# writing newline character
file1.write("\n")
file1.write("Today")

# without newline character
file1.write("Tomorrow")


file1 = open("myfile.txt", "r")
print("Output of Readlines after appending")
print(file1.read())
print()
file1.close()
```

- **OUTPUT:**

```
Output of Readlines after appending
This is Delhi
This is Paris
This is London
TodayTomorrow
```

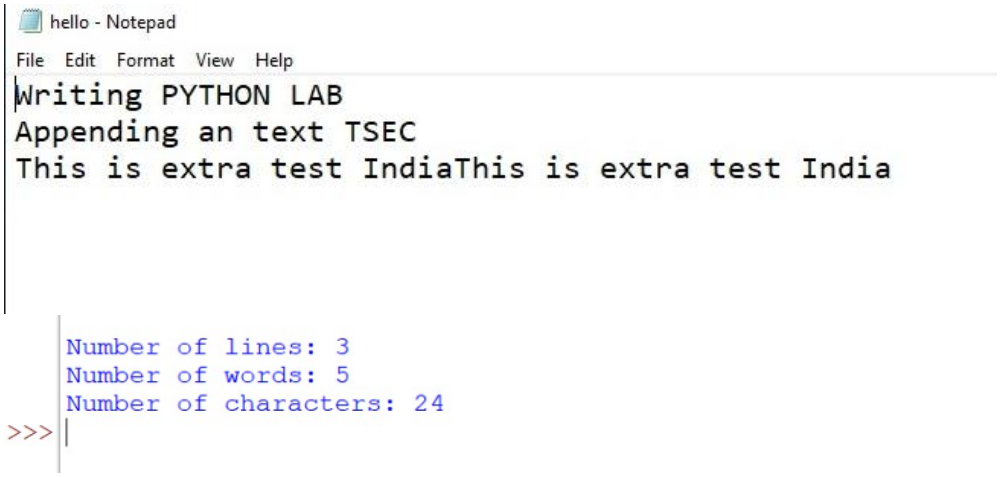
b) Count Number of Lines, Words, and Characters in a File:

To count the number of lines, words, and characters in a file, you can read the file and then use Python string and list methods to perform the required counts. For example, to count lines, you can split the content based on newline characters (\n). To count words, you can use the split() method, and to count characters, you can use the len() function.

- **PROGRAM 1:**

```
filename= r"C:\Users\Lenovo\Desktop\S21_2201072\hello.txt"
num_lines = 0
num_words = 0
num_chars = 0

with open(filename, 'r') as file:
    for line in file:
        num_lines += 1

words = line.split()
num_words += len(words)
num_chars += len(line)

file =  open(filename, 'a')
extra_text = "This is extra test India"

file.write(extra_text)
file.close()

print(f"Number of lines: {num_lines}")
print(f"Number of words: {num_words}")
print(f"Number of characters: {num_chars}")
```

- **OUTPUT:**



c) Display Files Available in the Current Directory

To display the files available in the current directory, you can use the os module, which provides a function called listdir(). This function returns a list containing the names of the entries in the directory given by the path. You can iterate over the list and print the names of the files.

What is a Directory in Python?

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

How to List Files in a Directory in Python

There are multiple ways of listing all the files in a directory. In this article, we will discuss the below modules and their functions to fetch the list of files in a directory. We will cover a total of 5 ways with examples to check the list of files in a directory.

1. Using **OS Module**
2. Using **glob Module**

Using os.listdir() method to get the list of files

os.listdir() method gets the list of all files and directories in a specified directory. By default, it is the current directory. Beyond the first level of folders, os.listdir() does not return any files or folders.

Syntax: *os.listdir(path)*

Parameters:

- ***Path***: *Path of the directory*

Return Type: *returns a list of all files and directories in the specified path*

Using os.walk() method to access files in a Directory tree

OS.walk() generates file names in a directory tree. This function returns a list of files in a tree structure. The method loops through all of the directories in a tree.

Syntax: *os.walk(top, topdown, onerror, followlinks)*

Parameters:

- ***top:*** *It is the top directory from which you want to retrieve the names of the component files and folders.*
- ***topdown:*** *Specifies that directories should be scanned from the top down when set to True. If this parameter is False, directories will be examined from the top down.*
- ***onerror:*** *It provides an error handler if an error is encountered*
- ***followlinks***: *if set to True, visits folders referenced by system links*

Return: *returns the name of every file and folder within a directory and any of its subdirectories.*

Using os.scandir() method to list files in a Directory

os.scandir() is an efficient version of os.listdir() function. It was later released by Python and is supported for Python 3.5 and greater.

Syntax: *os.scandir(path)*

Parameter:

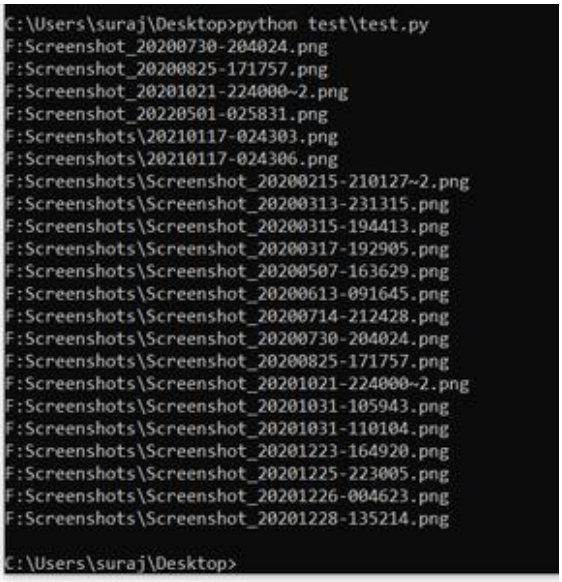
- ***Path-*** *Path of the directory.*

Return Type: *returns an iterator of os.DirEntry object.*

- **PROGRAM 1:**

```
import os
# traverse whole directory
for root, dirs, files in os.walk(r'F:'):
        # select file name
        for file in files:
                # check the extension of files
               if file.endswith('.png'):
                        # print whole path of files
                          print(os.path.join(root, file))
```

- **OUTPUT:**

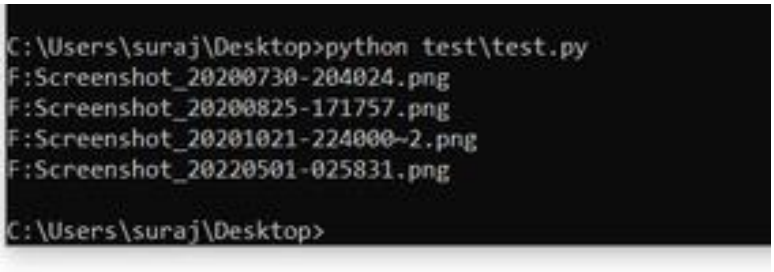


- **PROGRAM 2:**

```
import os
# return all files as a list
 for file in os.listdir(r'F:'):
            # check the files which are end with specific extension
              if file.endswith(".png"):
                        # print path name of selected files
```

```
print(os.path.join(r'F:', file))
```

- **<u>OUTPUT:</u>**



- ***<u>CONCLUSION:</u>*** **Hence, we have successfully implemented program on exploring files and directories; LO 2.**