Student Name: _____ **Roll No:** _____ **Section:** _____

Lab No. 10

Lab 10– File handling in python read, write and delete files

The file handling plays an important role when the data needs to be stored permanently into the file. A file is a named location on disk to store related information. We can access the stored information (non-volatile) after the program termination.

The file-handling implementation is slightly lengthy or complicated in the other programming language, but it is easier and shorter in Python.

In Python, files are treated in two modes as text or binary. The file may be in the text or binary format, and each line of a file is ended with the special character.

Hence, a file operation can be done in the following order.

- Open a file
- Read or write - Performing operation
- Close the file

The key function for working with files in Python is the **open()** function.

The **open()** function takes two parameters; filename, and mode.

There are four different methods (modes) for opening a file:

1. "r" - Read - Default value. Opens a file for reading, error if the file does not exist
2. "a" - Append - Opens a file for appending, creates the file if it does not exist
3. "w" - Write - Opens a file for writing, creates the file if it does not exist
4. "x" - Create - Creates the specified file, returns an error if the file exists

Student Name: **Roll No:** **Section:** ____ _ _

Character	Function
r	Open file for reading only. Starts reading from beginning of file. This default mode.
rb	Open a file for reading only in binary format. Starts reading from beginning of file.
r+	Open file for reading and writing. File pointer placed at beginning of the file.
w	Open file for writing only. File pointer placed at beginning of the file. Overwrites existing file and creates a new one if it does not exists.
wb	Same as **w** but opens in binary mode.
w+	Same as **w** but also alows to read from file.
wb+	Same as **wb** but also alows to read from file.
a	Open a file for appending. Starts writing at the end of file. Creates a new file if file does not exist.
ab	Same as **a** but in binary format. Creates a new file if file does not exist.
a+	Same a **a** but also open for reading.
ab+	Same a **ab** but also open for reading.

Python File Open

Syntax:

To open a file for reading it is enough to specify the name of the file:

```
#opens the file file.txt in read mode
fileptr = open("file.txt","r")
```

Open a file on a different location:

```
f = open("D:\\myfiles\geek.txt", "r")
print(f.read())
```

Student Name: _____ **Roll No:** _____ **Section:** _____

The close() method

Once all the operations are done on the file, we must close it through our Python script using the close() method. Any unwritten information gets destroyed once the close() method is called on a file object.

We can perform any operation on the file externally using the file system which is the currently opened in Python; hence it is good practice to close the file once all the operations are done.

The syntax to use the close() method is given below.

```
# opens the file file.txt in read mode
fileptr = open("file.txt","r")

if fileptr:
    print("file is opened successfully")

#closes the opened file
fileptr.close()
```

Read the File

```
f = open("geek.txt", "r")
print(f.read())
```

Writing the file

To write some text to a file, we need to open the file using the open method with one of the following access modes.

w: It will overwrite the file if any file exists. The file pointer is at the beginning of the file.

a: It will append the existing file. The file pointer is at the end of the file. It creates a new file if no file exists.

Consider the following example**.**

```
fileptr = open("geek.txt", "w")

# appending the content to the file
fileptr.write('''''Python is the modern day language.''')
```

Student Name: _____ Roll No: _____ Section: _____

```
# closing the opened the file
fileptr.close()
```

Example

```
f = open("geek.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("geek.txt", "r")
print(f.read())
```

Create a New File

To create a new file in Python, use the open () method, with one of the following parameters:

1. "x" - Create - will create a file, returns an error if the file exist
2. "a" - Append - will create a file if the specified file does not exist
3. "w" - Write - will create a file if the specified file does not exist

```
# Create a file called "myfile.txt"
f = open("myfile.txt", "x")
```

Read Lines of the file

Python facilitates to read the file line by line by using a function readline() method. The readline() method reads the lines of the file from the beginning, i.e., if we use the readline() method two times, then we can get the first two lines of the file.

Student Name: _____ _____ _____ **Roll No:** _____ _____ **Section:** _____ _____

Example 1:
Reading lines using readline() function

```
fileptr = open("geek.txt","r");
#stores all the data of the file into the variable content
content = fileptr.readline()
content1 = fileptr.readline()
#prints the content of the file
print(content)
print(content1)
#closes the opened file
fileptr.close()
```

We called the readline() function two times that's why it read two lines from the file.

Python provides also the readlines() method which is used for the reading lines. It returns the list of the lines till the end of file(EOF) is reached.
Example

```
#open the file.txt in read mode. causes error if no such file exists.
fileptr = open("geek.txt","r");

#stores all the data of the file into the variable content
content = fileptr.readlines()

#prints the content of the file
print(content)

#closes the opened file
fileptr.close()
```

Read file through for loop

```
fileptr = open("geek.txt","r");
#running a for loop
for i in fileptr:
    print(i) # i contains each line of the file
```

Student Name: _____ _ _ _ _ _ _ _ **Roll No:** _____ _ _ _ **Section:** _____ _ _

Python OS module

Renaming the file

The Python os module enables interaction with the operating system. The os module provides the functions that are involved in file processing operations like renaming, deleting, etc. The os module provides the **remove()** method which is used to remove the specified file. The syntax to use the **remove()** method is given below.

Syntax:

```
remove(file-name)
```

Removing the file

```
import os;
#deleting the file
os.remove("geek.txt")
```

```
import os
if os.path.exists("geek.txt"):
  os.remove("geek.txt")
else:
  print("The file does not exist")
```

Programming Exercise

Task 1:

Write a python program that to merge the content of two files in a new file

Task 2:

Write a python program that merge the content of one files in a new file