File handling:
Files, File Operations, Files and Streams, Creating a File, Reading From a File, Iterating Through Files, Writing file, Maintain students record using file handling

Till now, we were taking the input from the console and writing it back to the console to interact with the user.

Sometimes, it is not enough to only display the data on the console. The data to be displayed may be very large, and only a limited amount of data can be displayed on the console, and since the memory is volatile, it is impossible to recover the programmatically generated data again and again.

However, if we need to do so, we may store it onto the local file system which is volatile and can be accessed every time. Here, comes the need of file handling.

Here, we will learn all about file handling in python including, creating a file, opening a file, closing a file, writing and appending the file, etc.

Files are a way to save data permanently. Everything you've learned so far is resident only in memory; as soon as you close down Python or turn off your computer, it goes away. You would have to retype everything over if you wanted to use it again.

Python provides the open() function which accepts two arguments, file name and access mode in which the file is accessed. The function returns a file object which can be used to perform various operations like reading, writing, etc.

Opening a file

Python provides the open() function which accepts two arguments, file name and access mode in which the file is accessed. The function returns a file object which can be used to perform various operations like reading, writing, etc.

The syntax to use the open() function is given below.

file object = open(<file-name>, <access-mode>)

The files can be accessed using various modes like read, write, or append. The following are the details about the access mode to open a file.

SN	Access mode	Description
1	r	It opens the file to read-only. The file pointer exists at the beginning. The file is by default open in this mode if no access mode is passed.
2	rb	It opens the file to read only in binary format. The file pointer exists at the beginning of the file.
3	r+	It opens the file to read and write both. The file pointer exists at the beginning of the file.
4	rb+	It opens the file to read and write both in binary format. The file pointer exists at the beginning of the file.
5	w	It opens the file to write only. It overwrites the file if previously exists or creates a new one if no file exists with the same name. The file pointer exists at the beginning of the file.
6	wb	It opens the file to write only in binary format. It overwrites the file if it exists previously or creates a new one if no file exists with the same name. The file pointer exists at the beginning of the file.
7	w+	It opens the file to write and read both. It is different from r+ in the sense that it overwrites the previous file if one exists whereas r+ doesn't overwrite the previously written file. It creates a new file if no file exists. The file pointer exists at the beginning of the file.
8	wb+	It opens the file to write and read both in binary format. The file pointer exists at the beginning of the file.
9	a	It opens the file in the append mode. The file pointer exists at the end of the previously written file if exists any. It creates a new file if no file exists with the same name.
10	ab	It opens the file in the append mode in binary format. The pointer exists at the end of the previously written file. It creates a new file in binary format if no file exists with the same name.
11	a+	It opens a file to append and read both. The file pointer remains at the end of the file if a file exists. It creates a new file if no file exists with the same name.

12	ab+	It opens a file to append and read both in binary format. The file pointer remains at the end of the file.

```
fptr = open("sample.txt","r")
print(fptr)

if fptr:
   print("success")
fptr.close()
```

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 in the file system is the file is 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.

fileobject.close()

Reading the file

To read a file using the python script, the python provides us the read() method. The read() method reads a string from the file. It can read the data in the text as well as binary format.

The syntax of the read() method is given below.

fileobj.read(<count>)

Here, the count is the number of bytes to be read from the file starting from the beginning of the file. If the count is not specified, then it may read the content of the file until the end.

```
fptr = open("sample.txt","r")

y = fptr.read(10)
```

```
print(y)


x = fptr.read()

print(x)
```

Read Lines of the file

Python facilitates us to read the file line by line by using a function readline(). 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.

Consider the following example which contains a function readline() that reads the first line of our file "file.txt" containing three lines.

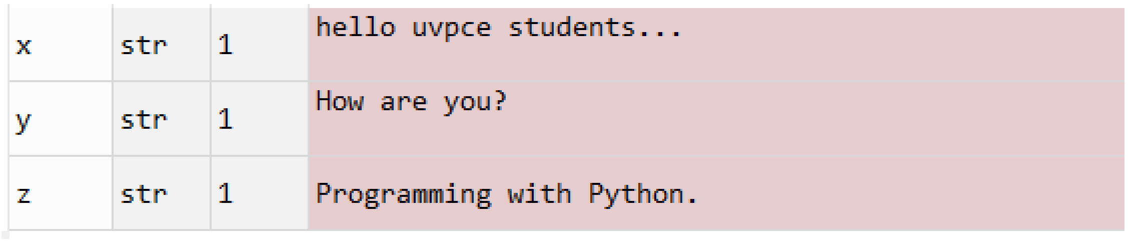
```
fptr = open("sample1.txt","r")


x = fptr.readline()

y = fptr.readline()

z = fptr.readline()


fptr.close()
```



x	str	1	hello uvpce students...
y	str	1	How are you?
z	str	1	Programming with Python.

Looping through the file

By looping through the lines of the file, we can read the whole file.

Example

```
fptr1 = open("sample1.txt","r")

for x in fptr1:

   print(x)
```

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.

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.

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

```
fptr = open("sample2.txt","w")

fptr.write("Python is a simple, high level, dynamic, object oriented and interpreted language")


fptr = open("sample2.txt","r")

x = fptr.read()

print(x)

fptr.close()


y = fptr.read()


fptr = open("sample2.txt","a")

fptr.write("append example")

fptr.close()
```

Creating a new file

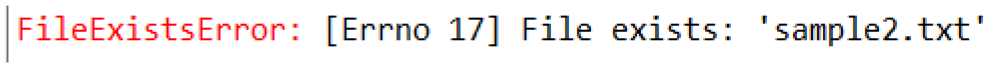
The new file can be created by using one of the following access modes with the function open(). x: it creates a new file with the specified name. It causes an error a file exists with the same name.

a: It creates a new file with the specified name if no such file exists. It appends the content to the file if the file already exists with the specified name.

w: It creates a new file with the specified name if no such file exists. It overwrites the existing file.

Consider the following example.

```
fptr = open("sample2.txt","x")
```



```
FileExistsError: [Errno 17] File exists: 'sample2.txt'
```

File Pointer positions

Python provides the tell() method which is used to print the byte number at which the file pointer exists. Consider the following example.

Example

```
fptr = open("sample.txt","r")

print("The pointer is at position : "+str(fptr.tell()))


x = fptr.read()

print("After - The pointer is at position : "+str(fptr.tell()))
```

Modifying file pointer position

In the real world applications, sometimes we need to change the file pointer location externally since we may need to read or write the content at various locations.

For this purpose, the python provides us the **seek()** method which enables us to modify the file pointer position externally.

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

```
<file-ptr>.seek(offset, from)
```

The seek() method accepts two parameters:

offset: It refers to the new position of the file pointer within the file.

from: It indicates the reference position from where the bytes are to be moved. If it is set to 0, the beginning of the file is used as the reference position. If it is set to 1, the current position of the file pointer is used as the reference position. If it is set to 2, the end of the file pointer is used as the reference position.

Consider the following example.

fptr = open("sample.txt","r")

print("current pointer position is: "+(str)(fptr.tell()))

fptr.seek(10)

print("current pointer position is: "+(str)(fptr.tell()))

<u>CSV File</u>

A csv stands for "comma separated values", which is defined as a simple file format that uses specific structuring to arrange tabular data. It stores tabular data such as spreadsheet or database in plain text and has a common format for data interchange. The csv file opens into the excel sheet, and the rows and columns data define the standard format.

Writing csv files

We can also write any new and existing csv files in Python by using csv.writer() module. It is similar to csv.reader() module and also has two methods, i.e. writer function or the Dict Writer class.

It presents two functions, i.e., **writerow()** and **writerows()**. The **writerow()** function only write one row and the **writerows()** function write more than one row.

Let's write following data to a CSV File.

```
data = [{'Rank': 'B', 'first_name': 'Parker', 'last_name': 'Brian'},
        {'Rank': 'A', 'first_name': 'Smith', 'last_name': 'Rodriguez'},
        {'Rank': 'C', 'first_name': 'Tom', 'last_name': 'smith'},
        {'Rank': 'B', 'first_name': 'Jane', 'last_name': 'Oscar'},
        {'Rank': 'A', 'first_name': 'Alex', 'last_name': 'Tim'}]
```

```
import csv

with open("students.csv","w") as csvfile:
   field_names = ['name','semester','branch','result']
   x = csv.DictWriter(csvfile,fieldnames=field_names)
```

```
    x.writeheader()
    x.writerow({'name':'Karan','semester':4,'branch':'CE','result':9.5})

    x.writerows([{'name':'Divyesh','semester':6,'branch':'CE','result':9.0},
                 {'name':'Hardik','semester':4,'branch':'IT','result':7.5},
                 {'name':'Naresh','semester':4,'branch':'ME','result':8.5}])

print('Finish')

import pandas as pd
df = pd.read_csv('students.csv')
print(df)
```