As Per CBSE Syllabus 2022-23

Computer Science

Class XII

Chapter-3

File handling In Python

What Is File Handling In Python?

Files are named location on the disk used to store data permanently. Files are non-volatile in nature.

Python offers a wide range of methods to handle the files stored on the disk, we can perform various operation like reading, writing, editing files stored in computer permanent storage though python programs.

File Handling Steps

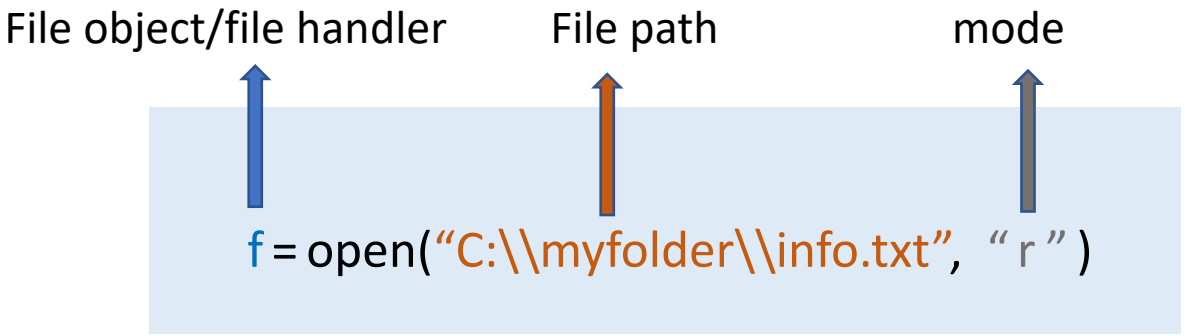
Step1-Open the File
Step 2-Process the file (Read/write/append)
Step 3-Close the file

Opening a file:

To perform any kid of operation on a file, the programmer first need to open the file by using open() method.

Syntax:

<file_object_name>=open(<filename>,<mode>)



Open() Method

File Pointer: File pointer/file handler is reference to the file on the disk which is being opened.

File Path: The Location of the file on the disk

Mode: This specifies the nature of operation(read/write/append) to be performed on the file.

Text File Mode	Binary File Mode	Description
'r'	'rb'	Read
'w'	'wb'	Write
'a'	'ab'	Append
'r+'	'rb+'	Read and write
'w+'	'wb+'	Write and read
'a+'	'ab+'	Write and read

File Access Modes

File Access Modes

Read Mode: when a file is opened in read mode then the file must exist, otherwise python will raise an I/O error.

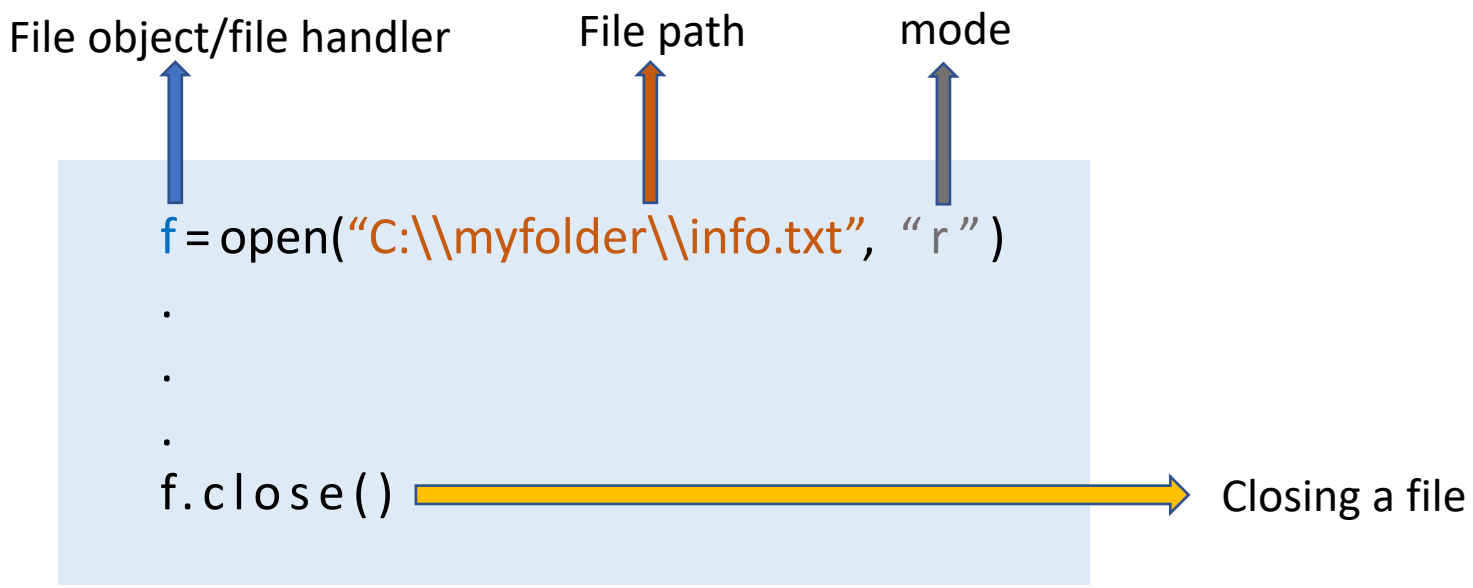
Write Mode: When a file is opened in write mode and if the file does not exist then a new file will be created at the specified location on the disk with the same name. If the file exist on the disk, then the file will be truncated, the user can overwrite new data into the file.

Append Mode: When a file is opened in append mode and if the file does not exist then a new file will be created at the specified location on the disk with the same name. If the file exist on the disk, then the file's exixting data will be retained and new data can be appended at the end of the file.

Closing a File

The close() method is used to close an opened file, the execution of close() method breaks the reference of the file object to file on disk, so after execution of close() method no operation on the file can be performed, without reopening the file.

Syntax:
<fileHandle>.close()



File Handling and File Types

Python offers wide range of modules/methods to perform file handling operation on both Text File and Binary File

Text File:

Text files stores the information on the file as sequence of ASCII or Unicode characters. These files can be open in any text editor in a human readable form.

Binary File:

Binary files stores information as a sequence of bytes, Binary files stores the information in the same format as it is stored in the memory.

Text File Handling:

Reading a Text File:

We can use any of these 3 methods/functions In order to perform read operation on a text file:

read([n])

readline([n])

readlines()

read method():

This function is used to read contents of a text file and returns the result as a string. If invoked with argument n it returns n number of bytes(characters) from file.

Syntax:

<file_object>.read([n])

```
#program to read a text file
f=open("C:\\myfolder\\cs.txt", "r")
data=f.read()
print(data)
f.close()
```

CS.txt File

Computing is part of everything we do. Computing drives innovation in engineering, business, entertainment, education and it provides solutions to complex, challenging problems.

Output:

Computing is part of everything we do. Computing drives innovation in engineering, business, entertainment, education and it provides solutions to complex, challenging problems.

```
#program to read first n characters of a text file
n=int(input("Enter no of chracetr you want to read"))
f=open("C:\\myfolder\\cs.txt", "r")
data=f.read(n)
print(data)
f.close()
```

Reads n bytes/characters from file

readline() method:

This function is used to read a line of input text file and returns the result as a string. If argument n specified then reads at n bytes(characters) of the current line.

Syntax:

<file_object>.readline([n])

```
#program to read first 2 lines of a text file
f=open("C:\\myfolder\\cs.txt", "r")
line1=f.readline()
print(line1)
line2=f.readline()
print(line2)
f.close()
```

cs.txt File

Computing is part of everything we do. Computing drives innovation in engineering, business, entertainment, education and it provides solutions to complex, challenging problems.

Output:

Computing is part of everything we do. Computing drives innovation in

readlines() method:

This function is reads all the line of the input text file and returns each line as string encapsulated inside a List.

Syntax:

<file_object>.readlines()

```
#program to read a text file using readlines()
f=open("C:\\myfolder\\cs.txt", "r")
data=f.readlines()
print(data)
f.close()
```

cs.txt File

Computing is part of everything we do. Computing drives innovation in engineering, business, entertainment, education and it provides solutions to complex, challenging problems.

Output:

['Computing is part of everything we\n', 'do. Computing drives innovation in\n', 'engineering, business, entertainment,\n', 'education and it provides solutions to \n', 'complex, challenging problems. \n']

Writing into a Text File:

Python provides these two functions for writing into a text file

Function Name	Syntax	Description
write()	<fileHandle>.write(str)	Writes the string argument str into the file
writelines()	<fileHandle>.writelines(L)	Writes the List argument L, containing strings.

```
#program to write a string into a text file
f=open("C:\\myfolder\\new.txt","w")
f.write("Programming is fun")
f.close()
```

Write the string into the file file

```
#program to write multiple lines into a text file using writeline()
L=["programming is fun", "lets start writing python programs"]
f=open("C:\\myfolder\\new.txt","w")
f.writeline(L)
f.close()
```

Writes multiple strings from the List into the file file

Note: If the file do not exist in memory, a new file named new.txt will be created

Appending into a Text File:

When a file is opened in append mode, then new data can be added into the file without erasing the existing old data in the file, the new data will be appended at the end of the existing data. To open a file append mode, file mode need to be set as append "a" while executing open() method.

cs.txt File

Computing is part of everything we do.

```
#program to append a string into a text file
f=open("C:\\myfolder\\new.txt","a")
f.write("Lets start programming")
f.close()
```

append the string at the end of the file

Output:
Computing is part of everything we do.Lets start programming

Note: If the file do not exist in memory, a new file named new.txt will be created

Binary File Handling

Binary files can store more structed python objects like list, tuple, dictionary by converting these objects into byte stream(called serialization/pickling), these objects can brought back to original form while reading the file (called unpickling).

Pickle module is very useful when it come to binary file handling. We will use dump() and load() methods of pickle module to write and read data objects into binary files.

Writing into a binary file:

The dump() function of pickle module can be used to write an object into a binary file.

Syntax: pickle.dump(<object_to_be_witten>, <filehandle>)

Ex:

```
import pickle
f=open('csSubjects.dat', 'wb')
tup1=("AI","Networking","DBMS","Python")
Pickle.dump(tup1,f)
f.close()
```

Import the pickle module to use dump() method

Mode need to be 'wb', specifying write operation on binary file

dump() method writes the tuple tup1 into the file referenced by file handle

Reading a binary file

To read from a binary file the load() method of the pickle module can be used. This method unpickles the data from the binary file(convert binary stream into original object structure).

Syntax: <object>=pickle.load(<filehandle>)

```
import pickle
f=open('cs.dat','wb')
d={"name":"amit","roll":20,"mark":320}
pickle.dump(d,f)

header=Pickle.load(f)
sdata=pickle.load(f)
print("Name:",sdata[name])
print("Roll No:",sdata[roll])
print("Mark:",sdata[mark])
f.close()
```

Reading from binary file using load() method

Writing into Binary File (Example Program)

#program to write details(name, rollno, mark) of n number of students into a binary file

```
import pickle
f=open("C:\\myfolder\\students.dat","wb")
n=int(input("Enter how many students details you want to enter"))
for i in range(n):
        name=input("Enter students name:")
        roll=int(input("Enter students Roll No:"))
        mark=int(input("Enter students Mark:"))
        data={"Name":name,"RollNo":roll,"Mark":mark}
        f.dump(data,f)
print("Details of ",n," number of students saved successfully")
f.close()
```

Reading from Binary File (Example Program)

#Program to read the details of all students in the binary file Students.dat created in the last example

```
import pickle
f=open("C:\\myfolder\\students.dat","rb")
print("********Students Details*********")
while True:
        try:
                data=pickle.load(f)
                print("Name:",data["Name"])
                print("Roll Number:",data["RollNo"])
                print("Mark:",data["Mark"])
        except EOFError:
                print("Reached end of the file")
                break
f.close()
```

Performing Search operation on a Binary File (Example Program)

```
#Program to search the details of students  based on students Roll Number
import pickle
f=open("C:\\myfolder\\students.dat","rb")
sroll=int(input("Enter the Roll Number of the student"))
found=False
print("*******Student Details*********")
while True:
        try:
            data=pickle.load(f)
                if data["RollNo"]==sroll:
                        found=True
                        print("Name:",data["Name"])
                        print("Roll Number:",data["RollNo"])
                        print("Mark:",data["Mark"])
            except EOFError:
                    break
If found==False:
            print("Data not found")
f.close()
```

Append into a Binary File (Example Program)

#Program to append details of a new student into the binary file Student.dat

```
import pickle
f=open("C:\\myfolder\\students.dat","ab")
name=input("Enter students name:")
roll=int(input("Enter students Roll No:"))
mark=int(input("Enter students Mark:"))
data={"Name":name,"RollNo":roll,"Mark":mark}
f.dump(data,f)
f.close()
```

Update a Binary File (Example Program)

#Program to update mark of a student in the the binary file Student.dat

```
import pickle
f=open(`C:\\myfolder\\students.dat',`rb')
alldata=[]
sroll=int(input(`Enter the Roll Number of the student:'))
newmark=int(input(`Enter the Mark to be updated: '))
while True:
        try:
              sdata=pickle.load(f)
              alldata.append(sdata)
        except EOFError:
               break
f.close()
for record in alldata:
        if record[`RollNo']==sroll:
             record[`Mark']=newMark
f=open(`C:\\myfolder\\students.dat',`rb')
for record in alldata:
       pickle.dump(record)
f.close()
```

Comma Separated File (CSV) File Handling

CSV stands for “Comma Separated Values.” It is used to store data in a tabular form in a text file. The fact that CSV files are actually text files, It becomes very useful to export high volume data to a database or to a server.

Reading/Writing operations on a CSV file can be performed using the csv module of python. to use the methods of csv module we have to import the csv module into our program.

```
import csv
```

Structure of a csv File

```
Name,Class,Section,Mark
Amit,12,A,460
Rahul,12,A,490
Priya,12,B,360
Sabir,12,B,440
Anjali,11,B,350
Jasprit,11,B,390
Ronit,12,A,480
```

Writing into a csv File

As csv file is basically a text file, to write into a csv file mode “w” must be specified along with the file path in the open() method to open the file.

f=open(`C:\\myfolder\\new.csv’,`w’)

After the the file is being open and a file handle is created, we need to create a writer object using writer() method.

<writerObject>=csv.writer(<fileHandle>)

The writerow() method can be used to write a List object into the csv file.

<writerObject>.writerow(<[List]>)

#program to write four rows into csv file

```
import csv
f=open(`C:\\myfolder\\new.csv’,`w’)
writer1=csv.writer(f)
Writer1.writerow([`Name’,`Class’,`Mark’])
Writer1.writerow([`Amit’,12,450)
Writer1.writerow([`Sahil’,12,460])
Writer1.writerow([`Anjali’,11,410)
f.close()
```

Reading From a csv File

To read data from a csv file, the file need to opened in the same way as any text file
(Note: The file extension is .csv)
f=open(`C:\\myfolder\\new.csv',`r')
After the the file is being open and a file handle is created, we need to create a reader object using reader() method.
<readerObject>=csv.reader(<fileHandle>)
Once created, a reader object carries all data from the referenced csv file.

#program to read a csv file

```
import csv
f=open(`C:\\myfolder\\new.csv',`r')
reader1=csv.reader(f)
for record in reader1:
        print(record)
f.close()
```