Now that you are familiar with these reading functions' working, let us write some programs.

Program 5.1 Write a program to display the size of a file in bytes.

```
myfile = open(r'E:\poem.txt', "r")
str = myfile.read()
size = len(str)
print("Size of the given file poem.txt is")
print(size, "bytes")
```

Output

```
>>>
Size of the given file poem.txt is
387 bytes
```

Program 5.2 Write a program to display the number of lines in the file.

```
myfile = open(r'E:\poem.txt', "r")
s = myfile.readlines()
linecount = len(s)
print("Number of lines in poem.txt is", linecount)
myfile.close()
```

Output

```
>>>
Number of lines in poem.txt is 9
```

5.4.2 Writing onto Text Files

After working with file-reading functions, let us talk about the writing functions for data files available in Python. (See Table 5.3). Like reading functions, the writing functions also work on open files, *i.e.*, the files that are opened and linked via a *file-object* or *file-handle*.

Table 5.3 *Python Data Files – Writing Functions*

S.No.	Name	Syntax	Description
1.	write( )	<filehandle>.write(str1)	writes string *str1* to file referenced by *<filehandle>*
2.	writelines( )	<filehandle>.writelines(L)	writes all strings in list L as lines to file referenced by *<filehandle>*
The <filehandle> in above syntaxes is the *file-object* holding open file's reference			

Appending a File

When you open a file in "w" or *write mode*, Python overwrites an existing file or creates a non-existing file. That means, for an existing file with the same name, the earlier data gets lost. If, however, you want to write into the file while retaining the old data, then you should open the file in "a" or *append mode*. A file opened in *append mode* retains its previous data while allowing you to add newer data into. You can also add a plus symbol (+) with file read mode to facilitate reading as well as writing.

That means, in Python, writing in files can take place in following forms :

(*i*) In an existing file, while retaining its content

(*a*) if the file has been opened in append mode ("a") to retain the old content.

(*b*) if the file has been open in 'r+' or 'a+' modes to facilitate reading as well as writing.

(ii) to create a new file or to write on an existing file after truncating/ overwriting its old content

(a) if the file has been opened in write-only mode ("w")

(b) if the file has been open in 'w+' mode to facilitate writing as well as reading.

(iii) Make sure to use **close( )** function on *file-object* after you have finished writing as sometimes, the content remains in memory buffer and to force-write the content on file and closing, the link of *file-handle* from file, *close( )* is used.

Let us consider some examples now.

Code Snippet 8 *Create a file to hold some data*

```
fileout = open("Student.dat", "w")
for i in range(5) :
    name = input("Enter name of student :")
    fileout.write(name)
fileout.close()
```

The write() will simply write the content in file without adding any extra character.

It's important to use close()

The sample run of above code is as shown below :

```
>>>
Enter name of student : Riya
Enter name of student : Rehan
Enter name of student : Ronaq
Enter name of student : Robert
Enter name of student : Ravneet
```

Now you can see the file created in the same folder where this Python script/program is saved (see Fig. 5.3(*a*) below). However, if you want to create the file in specific folder then you must specify the file-path as per the guidelines discussed earlier.

Also, you can open the created file ("student.dat" in above case) in *Notepad* to see its contents. Fig. 5.3(*b*) shows the contents of file created through *code snippet 5*.

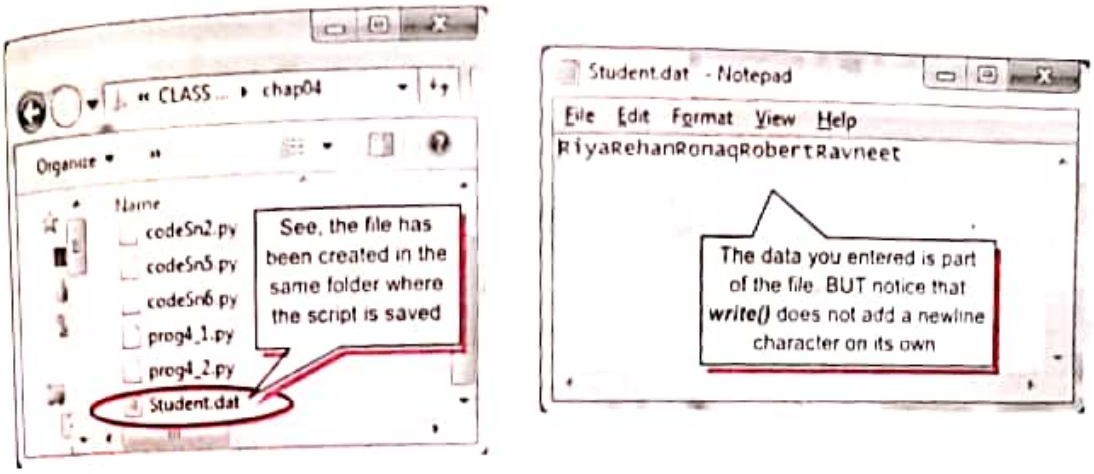


Figure 5.3 (*a*) File created through open( ) with "w" mode is stored in the same folder as that of script file (*b*) The write( ) function does not add any extra character in the file.

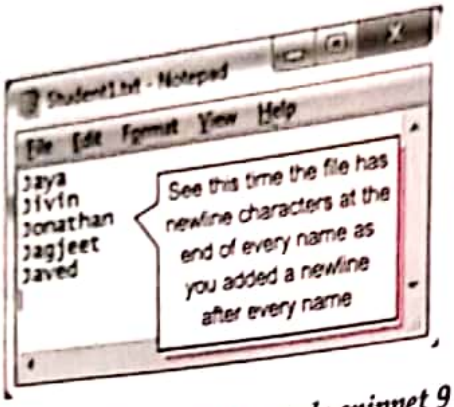
Carefully look at Fig. 5.3(b) that is showing contents of a file created through writing. It is clear from the file contents that *write( )* does not add any extra character like newline character('\n') after every write operation. Thus to group the contents you are writing in file, line wise, make sure to write newline characters on your own as depicted in *code snippet 9*.

Code Snippet 9 *Create a file to hold some data, separated as lines*

(This code is creating a different file than created with code snippet 9.)

```
fileout = open("Student1.txt", "w")
for i in range(5) :
    name = input("Enter name of student :")
    fileout.write(name)
    fileout.write('\n')
fileout.close()
```

The newline character '\n' written after every name



The sample run of the above code is as shown below :

```
>>>
Enter name of student : Jaya
Enter name of student : Jivin
Enter name of student : Jonathan
Enter name of student : Jagjeet
Enter name of student : Javed
```

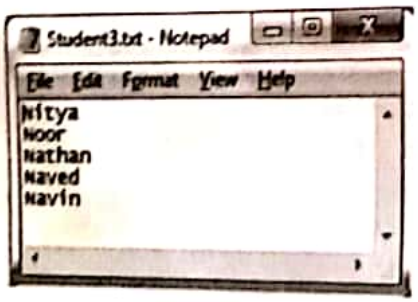
The file created by *code snippet 9* is shown above.

Code snippet 10 *Creating a file with some names separated by newline characters without using writelines() function.*

(For this, we shall use ***writelines( )*** *inplace of* ***write( )*** *function which writes the contents of a list to a file. Function writelines() also, does not add newline character, so you have to take care of adding newlines to your file.)*

```
fileout = open("Student3.txt", "w")
List1 = []
for i in range(5) :
    name = input("Enter name of student :")
    List1.append(name + '\n')
fileout.writelines(List1)
fileout.close()
```

Responsibility to add newline character is of programmer's



Sample run of the above code is as shown below

```
>>>
Enter name of student : Nitya
Enter name of student : Noor
Enter name of student : Nathan
Enter name of student : Naved
Enter name of student : Navin
```

Now that you are familiar with these writing functions' working, let us write some programs based on the same

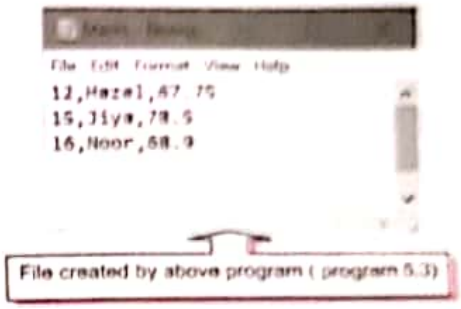
5.3 Write a program to get roll numbers, names and marks of the students of a class (get from user) and store these details in a file called "Marks.txt"

```
count = int(input("How many students are there in the class?"))
fileout = open("Marks.txt", 'w')
for i in range(count):
    print("Enter details for student", (i+1), "below:")
    rollno = int(input("Rollno:"))
    name = input("Name :")
    marks = float(input("Marks:"))
    rec = str(rollno) + "," + name + "," + str(marks) +'\n'
    fileout.write(rec)
fileout.close()
```

Joining individual information by adding commas in between

Output

```
How many students are there in the class? 3
Enter details for student 1 below :
Rollno : 12
Name : Hazel
Marks : 67.75
Enter details for student 2 below :
Rollno : 15
Name : Jiya
Marks : 78.5
Enter details for student 3 below :
Rollno : 16
Name : Noor
Marks : 68.9
```



File created by above program (program 5.3)

Figure 5.4 File created through program.

If you carefully notice, we have created comma separated values in one student record while writing in file. So we can say that the file created by program 5.4 is in a format similar to CSV (comma separated values) format or it is a delimited file where comma is the delimiter.

5.4 Write a program to add two more students' details to the file created in program 5.3.

```
fileout = open("Marks.txt", "a")
for i in range(2):
    print("Enter details for student", (i+1), "below :")
    rollno = int(input("Rollno:"))
    name = input("Name :")
    marks = float(input("Marks:"))
    rec = str(rollno) + "," + name + "," + str(marks) +'\n'
    fileout.write(rec)
fileout.close()
```

Notice the file is opened in append mode ("a") this time so as to retain old content

We want to add two records this time

```
>>>
Enter details for student 1 below :
Rollno : 17
Name : Akshar
Marks : 78.9
Enter details for student 2 below :
Rollno : 23
Name : Jivin
Marks : 89.5
```

```
12,Hazel,67.75
15,Jiya,78.5
16,Noor,68.9
17,Akshar, 78.9
23,Jivin,89.5
```

Same file after adding two more records

Program 5.5 Write a program to display the contents of file "Marks.txt" created through programs

```
fileinp = open("Marks.txt", "r")
while str :
    str = fileinp.readline()
    print(str)
fileinp.close()
```

Output

```
>>>
12,Hazel,67.75
15,Jiya,78.5
16,Noor,68.9
17,Akshar,78.9
23,Jivin,89.5
```

Have a look at some more examples of working with text files where we are using the following text file (Fig. 5.5)

File : *Answer.txt*

Letter 'a' is a wonderful letter.

It is impossible to think of a sentence without it.

We know this will never occur.

How mediocre our world would be without this single most powerful letter.

Figure 5.5 The contents of file Answer.txt.

Program 5.6 Write a program to read a text file line by line and display each word separated by a '#'.

```
myfile = open("Answer.txt", "r")
line = ""                          #initially stored a space (a non-None value)
while line :
    line = myfile.readline()  # one line read from file
    # printing the line word by word using split()
    for word in line.split():
        print(word, end = '#')
    print()
#close the file
myfile.close()
```

Output

```
Letter#'a'#is#a#wonderful#letter.#
It#is#impossible#to#think#of#a#sentence#without#it.#
We#know#this#will#never#occur.#
How#mediocre#our#world#would#be#without#this#single#most#powerful#letter.#
```