Chapter 5: File Handling

Introduction:

Most computer programs work with ***files***. This is because *files* help in storing information *permanently*. *Word processor* creates *document files*; *Database* programs create *files for information*; *Compilers* read source files and generate *executable files*. So we can see, it is the files that mostly worked with, inside programs.

A file in itself is a bunch of bytes stored on some storage device like hard etc.

Every programming language offers some provision to use and create *files* through *programs.* *Python* also allows us to create files through *data files*

Data Files:

The ***data files*** are the *files* that store data pertaining to a *specific application*, for later use. The data files can be stores in two ways; **Text Files** and **Binary Files**

i) Text Files:

A *text file* stores information in *ASCII or Unicode characters, (delimited)* with a special character known as *EOL (End of Line)* character. In *Python,* by default, this *EOL* character is the newline *('\n')* or *carriage – return*, newline combination *('\r\n').*

ii) Binary Files:

A ***binary file*** is just a file that contains information in the *same format* in which the information is *held in memory*, i.e., the file content that is *returned is raw* **(**with no translation or no specific encoding). In *binary file*, there is no *delimiter* for a *line*. Also no translations occur in *binary files*. As a result ***binary files*** are *faster and easier* for a program to read and write than are ***text files****.*

Opening and Closing Files:

In order to work with a *file* from within a Python program, we have to *open* it in a *specific mode* as per the *file manipulation tasks* we want to perform. The most basic file manipulation tasks including *adding, modifying or deleting data* in a *file,* which in turn include any one or combination of the following operations:

- *Reading data from files.*
- *Writing data to files.*
- *Appending data to files*.

Python provides **built-in functions** to perform each of these tasks.

Opening Files:

In *data file handling* through *Python*, in order to open a file we use *open() function* as per the following syntax:

<*file_objectname> =open(<filename>)*
<file objectname>=open(<filename>,<mode>)

For example:

myfile=open("info.txt")

The above opens *file "info.txt"* in *file mode* as *read mode* **(**because of *"r"* given as mode) and attached it to *file object namely myfile.*

Consider another statement:

file2=open ("data.txt","r")

The above statement *opens* the file *"data.txt"* in *read mode* **(**because of *"r"* given as mode) and attaches it to *file object* namely ***file2*****.**

Consider one more file-open statement:

file3=open ("e:\\main\\result.txt","w")

The above statement opens file *"result.txt"* ***(stored in folder E:\main)*** in *write mode* (because of *"w"* given as mode) and attaches it to file object namely ***file3.***

Note 1*:*
A file-object is also known as file-handle

Note 2:
The default file-open mode is read mode, i.e., if we do not provide any file open mode, Python will open it in read mode ("r").

There are ***two ways*** to give path in *filenames* correctly:

i) Double the *slashes e.g.,*

f = open("c:\\temp\\data.txt", "r")

ii) By giving *raw string* by *prefixing the file-path string* with an ***r*** *e.g.,*

f = open(r"c:\\temp\\data.txt", "r")

File Object/File Handle:

File objects are used to *read and write* data to a *file* on ***disk*****.** The *file object* is used to obtain a *reference* to the *file* on *disk* and open it for a *number of different tasks.*

File Access Modes:

When *Python* opens a *file*, it needs to know the *file-mode* in which the file *is being opened*. A *file-mode* governs the type of *operations (such as read or write or append)* possible in the *opened file* i.e., it refers to how the file will be used once it's opened. File modes supported by *Python* are as below:

Text file mode	**Binary File mode**	**Description**	**Notes**
'r'	**'rb'**	read only	File must exist already, otherwise Python raises I/O error.
'w'	**'wb'**	write only	i) If the file does not exist, file is created ii) If the file exists, Python will truncate existing data and overwrite in the file.
'a'	**'ab'**	Append	i) File is in write only mode ii) If the file exists, the data in the file is retained and new data being written will be appended to the end. iii) If the file does not exist, Python will create a new file
'r+'	**'r+b' or 'rb+'**	read and write mode	i) File must exists otherwise error is raised. ii) Both reading and writing operations can take place.

'w+'	**'w+b' or 'wb+'**	write and read	i) File is created if does not exist. ii) If file exists, file is truncated (past data is lost) iii) Both reading and writing operations can take place.
'a+'	**'a+b' or 'ab+'**	write and read	i) File is created if does not exist. ii) If file exists, file is **retained;** new data appendd iii) Both reading and writing operations can take place

Closing Files:

An opened file can be closed by calling the ***close()*** method of its *file-object*. A ***close()*** function *breaks* the link of *file-object* and the *file* on the disk. After *close(),* no tasks can be performed on that file through the ***file object (or file-handle).***
The general form of *close()* method is as follows:

<fileHandle>.close()

For instance, if a file *Master.txt* is opened ***via*** *file-handle* ***outfile,*** it may be closed by the following statement:

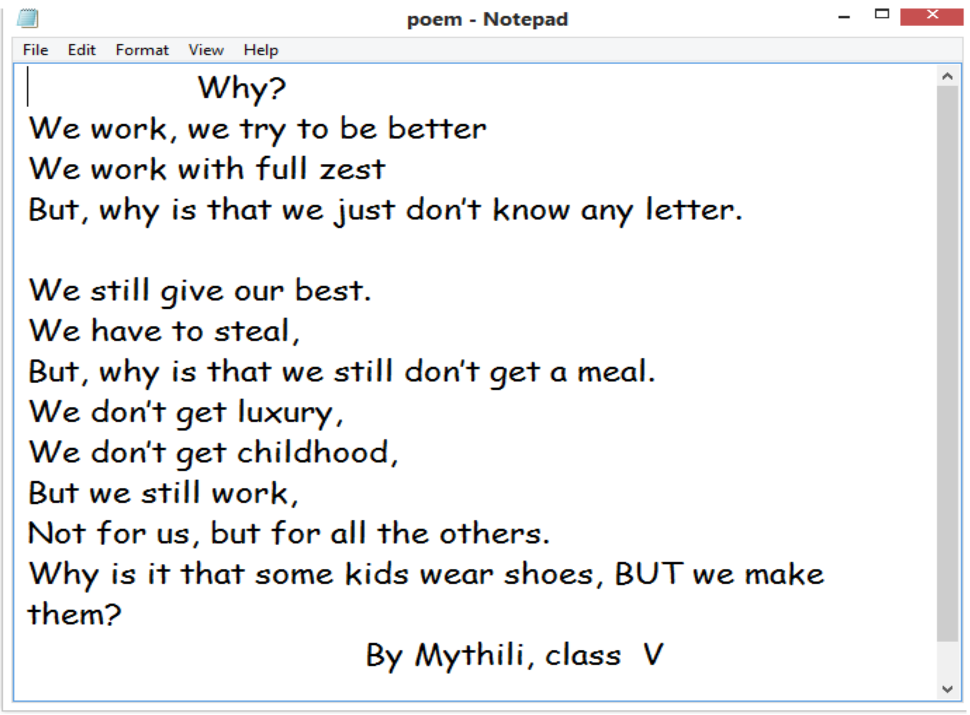
Outfile.close()

Reading from Files:

Python provides mainly *three types of read functions* to *read* from a data *file.*

Example:

Consider/ create a file poem.txt
(Assume the path of the file is: E\textfiles\poem.txt)



```
             Why?
We work, we try to be better
We work with full zest
But, why is that we just don't know any letter.

We still give our best.
We have to steal,
But, why is that we still don't get a meal.
We don't get luxury,
We don't get childhood,
But we still work,
Not for us, but for all the others.
Why is it that some kids wear shoes, BUT we make
them?
                         By Mythili, class  V
```

i) read():

Reads at most ***n bytes***; if no ***n*** is specified, reads the *entire file*.
This function *returns* the *bytes* in the form of a *string.*

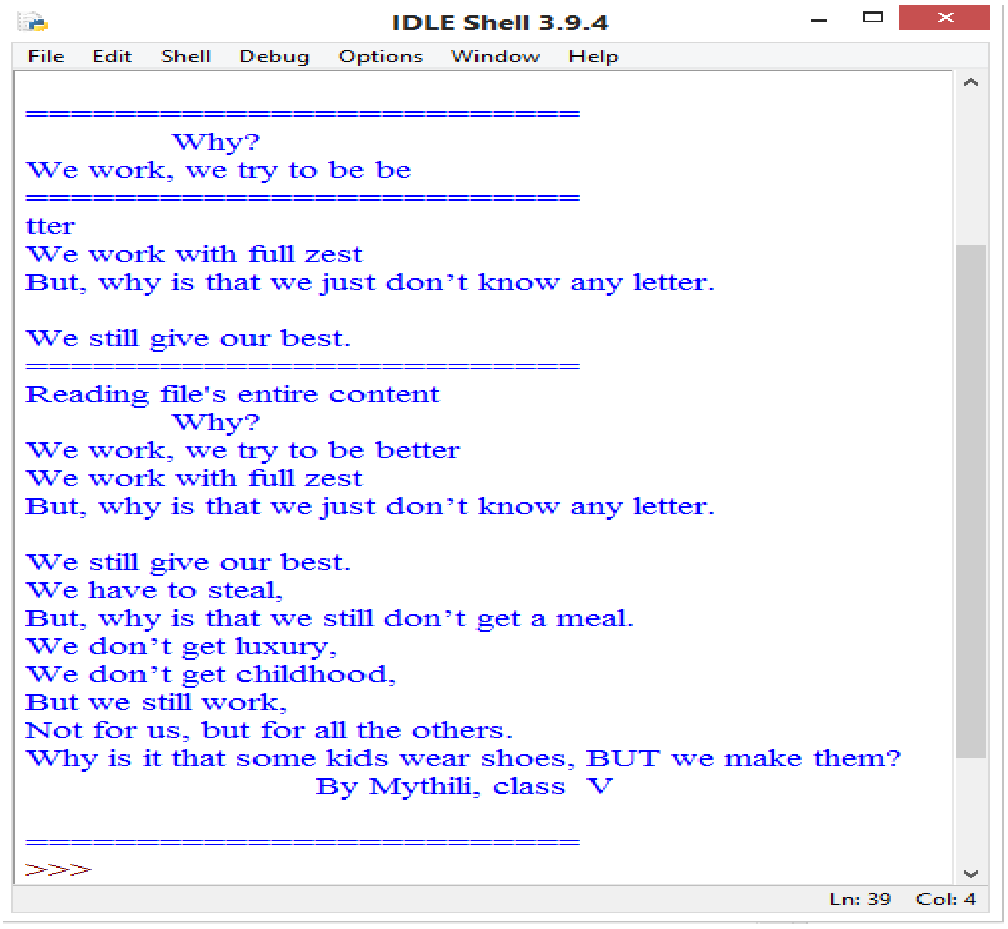
Syntax:
<filhandle>.read([n])

Example
#prg 1. Write a Python program to illustrate the use of read() method

```
print("\n========================")
myfile=open(r'E:\textfiles\poem.txt',"r")
str1=myfile.read(30)           # Reading 30 bytes
print(str1)
print("========================")
str2=myfile.read(100)          # Reading 100 bytes
print(str2)
myfile.close()
print("========================")

print("Reading file's entire content")
myfile=open(r'E:\textfiles\poem.txt',"r")
abc=myfile.read()              # Reading entire content
print(abc)
myfile.close()
print('========================')
```

Output:



```
==============================
              Why?
We work, we try to be be
==============================
tter
We work with full zest
But, why is that we just don't know any letter.

We still give our best.
==============================
Reading file's entire content
              Why?
We work, we try to be better
We work with full zest
But, why is that we just don't know any letter.

We still give our best.
We have to steal,
But, why is that we still don't get a meal.
We don't get luxury,
We don't get childhood,
But we still work,
Not for us, but for all the others.
Why is it that some kids wear shoes, BUT we make them?
                            By Mythili, class  V

==============================
>>>
```