So, what is a file? A file is a named location on a secondary storage media where data are permanently stored for later access.

Types of Files

Computers store every file as a collection of 0s and 1s i.e., in binary form. Therefore, every file is basically just a series of bytes stored one after the other. There are mainly two types of data files — text file and binary file. A text file consists of human readable characters, which can be opened by any text editor. On the other hand, binary files are made up of non-human readable characters and symbols, which require specific programs to access its contents.

Text file

A text file can be understood as a sequence of characters consisting of alphabets, numbers and other special symbols. Files with extensions like .txt, .py, .csv, etc. are some examples of text files. When we open a text file using a text editor (e.g., Notepad), we see several lines of text. However, the file contents are not stored in such a way internally. Rather, they are stored in sequence of bytes consisting of 0s and 1s. In ASCII, UNICODE or any other encoding scheme, the value of each character of the text file is stored as bytes. So, while opening a text file, the text editor translates each ASCII value and shows us the equivalent character that is readable by the human being. For example, the ASCII value 65 (binary equivalent 1000001) will be displayed by a text editor as the letter 'A' since the number 65 in ASCII character set represents 'A'.Each line of a text file is terminated by a special character, called the End of Line (EOL). For example, the default EOL character in Python is the newline (\n). However, other characters can be used to indicate EOL. When a text editor or a program interpreter encounters the ASCII equivalent of the EOL character, it displays the remaining file contents starting from a new line. Contents in a text file are usually separated by whitespace, but comma (,) and tab (\t) are also commonly used to separate values in a text file.

Binary Files

Binary files are also stored in terms of bytes (0s and 1s), but unlike text files, these bytes do not represent the ASCII values of characters. Rather, they represent the actual content such as image, audio, video, compressed versions of other files, executable files, etc. These files are not human readable. Thus, trying to open a binary file using a text editor will show some garbage values. We need specific software to read or write the contents of a binary file. Binary files are stored in a computer in a sequence of bytes. Even a single bit change can corrupt the file and make it unreadable to the supporting application. Also, it is difficult to remove any error which may occur in the binary file as the stored contents are not human readable. We can read and write both text and binary files through Python programs.

Opening and Closing a Text File

In real world applications, computer programs deal with data coming from different sources like databases, CSV files, HTML, XML, JSON, etc. We broadly access files either to write or read data from it. But operations on files include creating and opening a file, writing data in a file, traversing a file, reading data from a file and so on. Python has the io module that contains different functions for handling files.

Opening a file

To open a file in Python, we use the open() function. The syntax of open() is as follows:

file_object = open(file_name, access_mode)

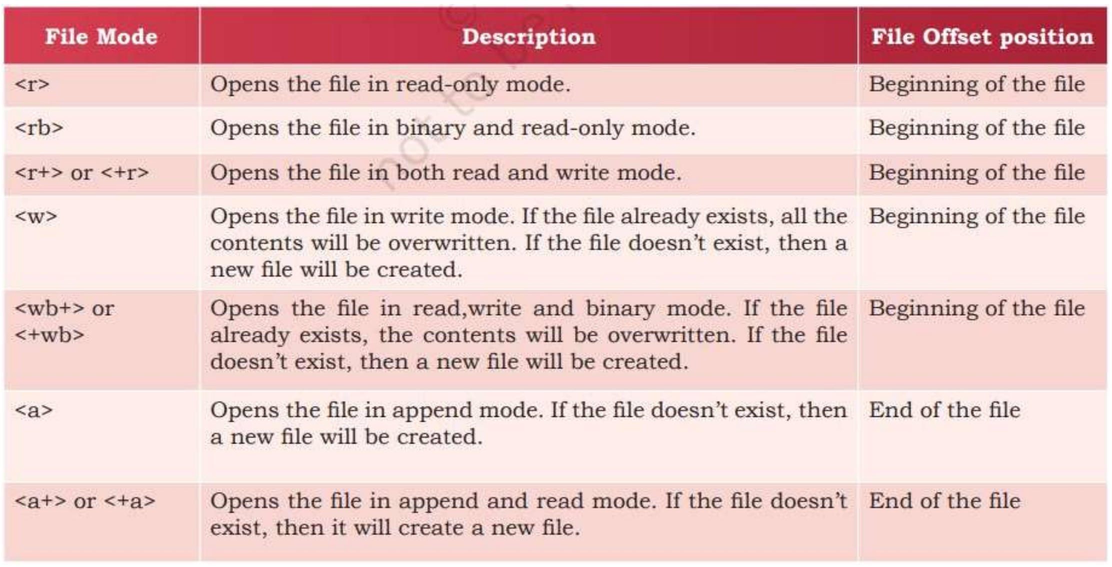
The file_object has certain attributes that tells us basic information about the file, such as:

• <file.closed> returns true if the file is closed and false otherwise.

• <file.mode> returns the access mode in which the file was opened.

• <file.name> returns the name of the file.

File Open Modes



File Mode	Description	File Offset position
<r>	Opens the file in read-only mode.	Beginning of the file
<rb>	Opens the file in binary and read-only mode.	Beginning of the file
<r+> or <+r>	Opens the file in both read and write mode.	Beginning of the file
<w>	Opens the file in write mode. If the file already exists, all the contents will be overwritten. If the file doesn't exist, then a new file will be created.	Beginning of the file
<wb+> or <+wb>	Opens the file in read,write and binary mode. If the file already exists, the contents will be overwritten. If the file doesn't exist, then a new file will be created.	Beginning of the file
<a>	Opens the file in append mode. If the file doesn't exist, then a new file will be created.	End of the file
<a+> or <+a>	Opens the file in append and read mode. If the file doesn't exist, then it will create a new file.	End of the file

Closing a file

Once we are done with the read/write operations on a file, it is a good practice to close the file. Python provides a close() method to do so. While closing a file, the system frees the memory allocated to it.

The syntax of close() is:

```
file_object.close()
```

Opening a file using with clause

In Python, we can also open a file using with clause.

The syntax of with clause is:

```
with open (file_name, access_mode) as file_object:
```

The advantage of using with clause is that any file that is opened using this clause is closed automatically, once the control comes outside the with clause. In case the user forgets to close the file explicitly or if an exception occurs, the file is closed automatically. Also, it provides a simpler syntax.

```
with open(“myfile.txt”,”r+”) as myObject:
    content = myObject.read()
```

Here, we don’t have to close the file explicitly using close() statement. Python will automatically close the file.