Files

File is the collection of data that is available to a program. We can retrieve and use data stored in a file whenever we required.

Advantages:-

- Stored Data is permanent unless someone remove it.
- Stored data can be shared.
- It is possible to update or remove the data.

Type of Files

There are two type of files:-

Text File – It stores data in the form of characters. It is used to store characters and strings.

Binary File – It stores data in the form of bytes, a group of 8 bits each. It is used to store text, images, pdf, csv, video and audio.

Text Mode and Binary Mode

Text Mode – A file opened in text mode, treats its contents as if it contains text strings of the str type.

When you get data from a text mode file, Python first decodes the raw bytes using either a platform-dependent encoding or, specified one.

Binary Mode – A file opened in Binary Mode, Python uses the data in the file without any decoding, binary mode file reflects the raw data in the file.

Opening a File

If we want to use a file or its data, first we have to open it.

open( ) – Open ( ) function is used to open a file. It returns a pointer to the beginning of the file. This is called file handler or file object.

Syntax:- open(‘filename’, mode='r', buffering, encoding=None, errors=None, newline=None, closefd=True, opener=None)

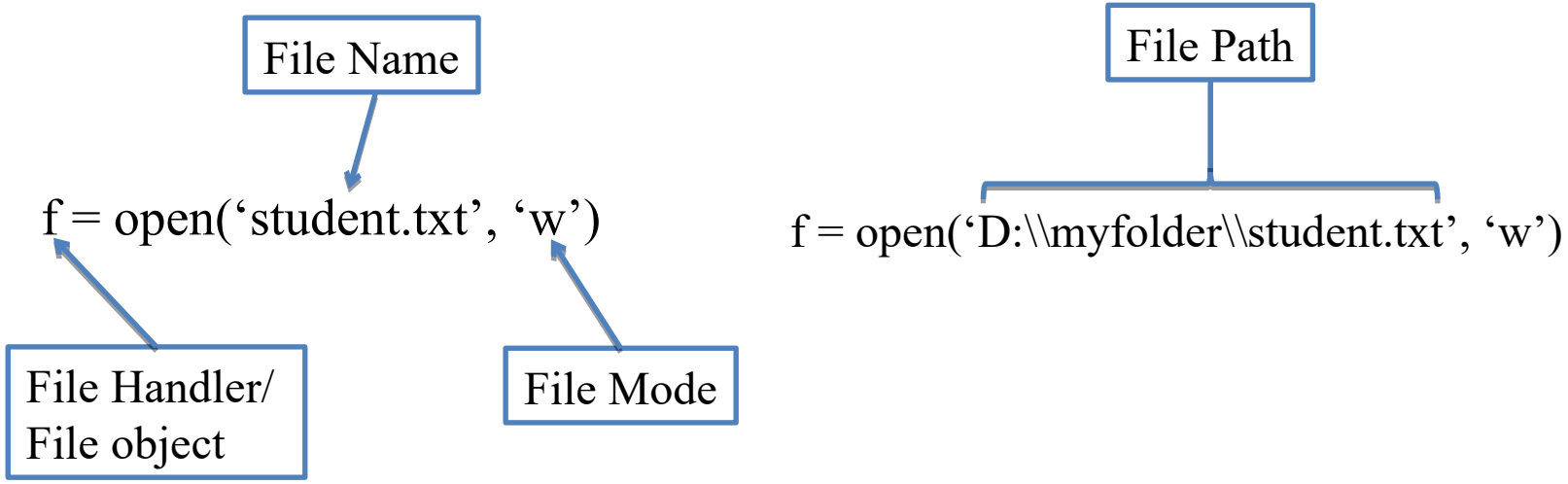
- filename – It represents a name of a file.
- mode – It represents the purpose of opening the file. It defaults to 'r' which means open for reading in text mode.
- buffering – It is an integer value used to set the size of the buffer for the file. In the binary mode we can pass 0 as buffering integer to inform not to use any buffering. In text mode we can pass 1 for buffering to retrieve data from the file one line at a time. We can pass any positive integer. Default is 4096 or 8192 bytes.

Opening a File

Syntax:- open(‘filename’, mode='r', buffering, encoding=None, errors=None, newline=None, closefd=True, opener=None)

- encoding – name of the encoding used to decode or encode the file. It should be used only in text mode. Ex:- utf-8
- errors – an optional string that specifies how encoding and decoding errors are to be handled, this cannot be used in binary mode. Some of the standard values are strict, ignore, replace etc.
- newline: this parameter controls how universal newlines mode works (it only applies to text mode). It can be None, ”, ‘\n’, ‘\r’, and ‘\r\n’.
- closefd – If closefd is False and a file descriptor rather than a filename was given, the underlying file descriptor will be kept open when the file is closed. If a filename is given closefd must be True (the default) otherwise an error will be raised.
- opener: A custom opener can be used by passing a callable as opener.

Opening a File



Text File Mode

Character	Meaning
r	Open for reading. The file pointer is positioned at the beginning of the file. If the file doesn't exist it will show FileNotFoundError.
w	Open for writing. If any data is already present in the file, it will overwrite the data. If the file doesn't exist it will create that file.
x	Open for exclusive creation with write. The specified file must not be available, if the specified file is available it will show error FileExistsError
a	Open for appending. The file pointer is positioned at the end of the file. It appends new data at the end of file. If the file does not exists it will create a new file for writing data.
r+	Open for reading and then writing
w+	Open for writing and then reading. It will overwrite existing data
a+	Open for appending then reading. It won't overwrite existing data

Binary File Mode

Character	Meaning
rb	Open for reading. The file pointer is positioned at the beginning of the file. If the file doesn’t exist it will show FileNotFoundError.
wb	Open for writing. If any data is already present in the file, it will overwrite the data. If the file doesn’t exist it will create that file.
xb	Open for exclusive creation with write. The specified file must not be available, if the specified file is available it will show error FileExistsError
ab	Open for appending. The file pointer is positioned at the end of the file. It appends new data at the end of file. If the file does not exists it will create a new file for writing data.
rb+	Open for reading and then writing
wb+	Open for writing and then reading. It will overwrite existing data
ab+	Open for appending then reading. It won't overwrite existing data

Closing a File

close( ) – This method is used to close, opened file.

Once we close the file, file object is deleted from the memory hence file will be no longer accessible unless we open it again.

If you don’t explicitly close a file, Python’s garbage collector will eventually destroy the object and close the open file for you, but the file may stay open for a while so You should always close opened file.

What will happened if we do not close opened file:-

- Data of the file may be corrupted or deleted.
- Memory utilized by the file is not freed it may cause of insufficient memory.

File object Variables

name – This shows the name of specified file.
Syntax:- file_object.name

mode – This shows mode (purpose) of the file.
Syntax:- file_object.mode

closed – This used to check whether file has closed or not.
It shows True if file is closed else shows False.
Syntax:- file_object.closed

File object Methods

readable() – This method is used to check whether file is readable or not.
It returns True if file is readable else returns False.
Syntax:- file_object.readable()

writable() – This method is used to check whether file is writable or not.
It returns True if file is writable else returns False.
Syntax:- file_object.writable()

Check File exists or not

isfile() – This method is used to check whether specified file is exists or not. This method belongs to path module which is sub module of os module.

Syntax:-

```
import os
os.path.isfile(filename)
```

Writing Data to the file

write ( ) – This method is used to store/write character or string into the file represented by the file object. It returns the number of character written.

Syntax:- file_object.write(string)

writelines ( ) – This method is used to store/write group of string (list, tuple, set) into the file represented by the file object.

Syntax:- file_object.writelines(group of string)

Reading Data from file

read (size) – This method is used to read data/content from a file and returns it as string in text mode or bytes object in binary mode.

Syntax:- file_object.read(size)

Where size represents the number of bytes to be read from the beginning of the file.

When size is omitted or negative, the entire contents of the file will be read and returned.

If the end of the file has been reached, file_object.read() will return an empty string (‘’).

Reading Data from file

readline () – This method is used to read single line from a file.

Syntax:- file_object.readline()

readlines () – This method is used to read all lines from a file. It will return list of line.

Syntax:- file_object.readlines()

Methods

tell ( ) - This method is used to find current position of file pointer from beginning of the file. Position starts from 0.

Syntax:- file_object.tell()

seek(position) – This method is used to move file pointer from one position to another position from beginning of the file. Position starts from 0 and it must be positive integer.

Syntax:- file_object.seek(position)

with Statement

The with statement can be used while opening a file.

When we open a file using with statement there is no need to close the file explicitly.

Syntax:-

```
with open (‘filename’, mode=‘r’) as file_object :
    statements
```

Ex:-

```
with open(‘student.txt’) as f :
    f.read()
```