How is the file stored in the Computer?:-

Computer store data in 0s & 1s, the binary form.

Every file is basically just a series of bytes stored one after the other.

There are mainly two types of files -> 1. Text file 2. binary file.

Text file:-

A text file consists of human-readable characters, which can be opened by any text editor.

A text file can be understood as a sequence of characters consisting of alphabets, numbers, and other special symbols.
some text file extensions are -> .txt, .py, .csv, etc.

You can open the file by using a text editor (e.g. Notepad)

The file does not store in a computer the way we see, it is stored in bytes consists 0s & 1s.

When we open a file text editor translates each ASCII value and shows us the equivalent character that is readable by the human being.

Ex:- The ASCII value 65 (BInary equivalent 1000001) that will be displayed by a text editor as the letter 'A'

Each line of a text file is terminated (Bring to an end) by a special character, called the End of Line (EOL).

The default Eol character in Python is the newline(\n).

Binary files:-

These files are made up of nonhuman-readable characters & symbols.

It requires specific programs to access its contents.

It stores data in terms of Bytes (0s & 1s).

It represents the actual content such as images, audio, video, compressed versions of other files, executable files, etc.

If we these files in text editore then we see garbage values.

We need specific software to read & write the content of the binary files.

Even a single bit change can corrupt the file and make it unreadable to the supporting application.

We can read and write both text & binary files through Python programs.

Opening a file in Python:-

To open a file in Python, we use the open() function. The syntax of open() is as follows:-
var_name = open("file name", "opening mode")

The function returns a file object called file handle which is stored in the variable --> var_name.

If the file does not exist, the above statement creates a new empty file and assigns it the name we specify int he statement.

---> Some file handling attributes:-

<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.

If the file is not in the current working directory then we need to specify the complete path of the file along with its name.

Default opening_mode is read mode.

Opening_mode ----------> Description

" r " --> Opens the file in read-only mode.

" rb " --> Opens the file in both read & read-only mode.

" r+ " & " +r "--> Opens the ifle in both read & write mode.

" w " --> Opens the file in write mode. If the file already exists, all the data overwrite(which means the old data will be removed and new data is stored). If the file doesn't exist, then a new file will make.

" w+ " & " + w " --> Open the file in read, write & binary mode.

" a " --> Opens the file in append mode(which means no old data will be removed). If the file doesn't exist, then a new file will be created.

" +a " & " a+ " --> Opens the file in append and read mode.

Closing a file:-

While closing a file, the system frees the memory allocated to it.

Syntax is;- file_object.close( )

Closing the file is necessary because if we try to access the file that is still open in another program or process, you may corrupt the file's data. so ensure to close the file.

Opening a file using the clause:-

The syntax with clause is;-
with open(file_name , opening_mode) as var_name:

If we use clause we don't worry about the closing of the file it closes the file automatically after the work is done.

Writing to a text file:-

To write data, we have to first open the file in write or append mode.

By using write mode, the old data of the file will be erased, and the cursor positioned at the beginning of the file.

In append mode, new data will be added at the end of previous data no old data is erased, and the cursor positioned at the end of the file.

You cannot write the numeric value(integer) as the parameter of the write function.

In write mode you only enter the string, if you enter any list or tuple, etc. into a file it shows an error.

So, to write a list, tuple, etc. in the file you have to use writelines ( ) function. writelines( ) function take any data type as an argument.

Write( ) => for writing a single string.
Writelines( ) => for writing a sequence of strings.

The read( ) method:-

This method is used to read a specified number of bytes of data from a data file.

The Syntax --> file_object.read( )

n is the no. of characters to read.

If it is empty then it read the whole file.

if there is any negative value of n is given

The readline([n]) method:-

This method reads one complete from the file separated by a newline(\n) character.

Syntax --> Myobject.readline(n)

n is the no. of characters in a line, and the maximum value of n is the no. of characters in the line.

When n is not given it read the whole line.

When the value of n is in negative then it also read the whole file.

The readlines( ) method.

This method reads all the lines and returns the lines along with the new line as a list of strings.

With readlines we see data in the form of a list.

Setting offset in a file:-

It is used to access data in a random fashion.

The tell( ) method:-

It is used to get the position of the pointer in the file.

This default shows 0 because the pointer starts from the beginning of the file.

The syntax --> File_object.tell( )

The seek( ) method:-

It is used to change the position of the file pointer within the file.

The file pointer represents the position in the file where the next read or write operation will take place.

The Syntax --> file_object.seek(no.of bytes, refrence_point).

value of reference point:-
0 means the beginning of the file (default).
1 means the current position of the file pointer.
2 means the end of the file.

The Pickle Module:-

The pickle module is used to save the object/data to get it later.

It is used for serializing & de-serializing any Python object structure.

Serialization is the process of transforming data or an object in memory (RAM) to a stream of bytes called Byte Streams.

Serialization also called pickling means converting any data to binary mode and vice-versa.

De-Serialization is also called unpickling means converting back binary data to normal data.

In the pickle module, data are not written but dumped, and similarly, data are not read but loaded.

The dump( ) method;:-

To dump the file we have to open the file in write mode(w).

It is used to convert data in binary mode.

Syntax of dump( ) --> pickle.dump(data_object, file_object)

The load( ) method:-

This is used to load (unpickling) data from a binary file.

The file is to be loaded in opened in binary read(rb) mode.

Syntax of load( ) --> Var = pickle. load(file-object).

Transient

It for a short time and produce some output, but when it ends, their data disappears. If you run the programs again, it starts with a clean slate. This happens because the data entered is executed inside primary memory, RAM, which is volatile (Temporary) in nature.

Persistent

It runs for a long time(or all the time); they keep at least some of its data in permanent storage(for ex:- a hard drive) and if it shut down or restarts, they execute from the same point. Examples of persistent programs are operating systems, which run whenever a computer is

switched on, and web servers, which run all the time, waiting for requests to come in on the network.

* when you open a file in read mode, the given file must exist in the folder, otherwise, Python will raise FileNOtFoundError.

CSV File

It stands for comma-separated values.

CSV is just like a test file, i.e., it is in a human-readable format.

It is used to store data in tabular form(in the form of rows and columns) like a spreadsheet or a database.

The separator character of CSB files is called a delimiter.

Its default delimiter is a comma(,).

Other delimiters which can be used with CSV files are tab ('\t'), colon (:), pipe(|), and semicolon(;).

It is the most preferred import and export format for databases and spreadsheets.

Each line in a CSV file is treated as a record.