UNIT V (GE8151 PROBLEM SOLVING AND PYTHON PROGRAMMING)

I. FILES

- File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk).
- Since, random access memory (RAM) is volatile which loses its data when computer is turned off, we use files for future use of the data.

There are many different types of files.

- Data File
- Text File
- Program File
- Directory File

Types of files

Python provides inbuilt functions for creating, writing and reading files.

There are two types of files that can be handled in python,

- Text files
- Binary files (written in binary language,0s and 1s).

1. Text Files:

- Text files are structured as a sequence of lines where each line includes a sequence of characters.
- Each line is terminated with a special character, called the End of Line character.

Examples for Text files:

- **Web standards:** html, XML, CSS, JSON etc.
- **Source code:** c, app, js, py, java etc.
- **Documents:** txt, tex, RTF etc.
- **Tabular data:** csv, tsv etc.
- **Configuration:** ini, cfg, reg etc.

2. Binary files

- All binary files follow a specific format. Because all the binary files will be encoded in the binary format, which can be understood only by a computer or machine.
- Binary files can only be processed by application.
- For handling such binary files we need a specific type of software to open it.

Binary files can be used to store text, images, audio and video.

Example:

- Executables
- Databases
- Application data
- Configuration files
- Device driver files

FILE OPERATIONS

When the user want to do some operations in the file there is a sequence of steps that should be followed.

- Open the File
- Read or Write data in the file(perform operation)
- Close the File

1. OPENING A FILE

The open() method

- Python has a built-in function open() to open a file. This function returns a file object and has two arguments which are file name & opening mode. The opening modes are reading, writing or appending.
- *Syntax:*

```
file_object=open("filename", "mode")
```

where file_object is the variable to add the object and the mode tells the interpreter which way the file will be used.

Example

```
f=open("E:/Python/text.txt", 'r')
f=open("E:/Python/text.txt", 'w')
```

FILE OPENING MODES

S.NO	Modes	Description
1	**r**	**Read Mode:** Read mode is used only to read data from the file.

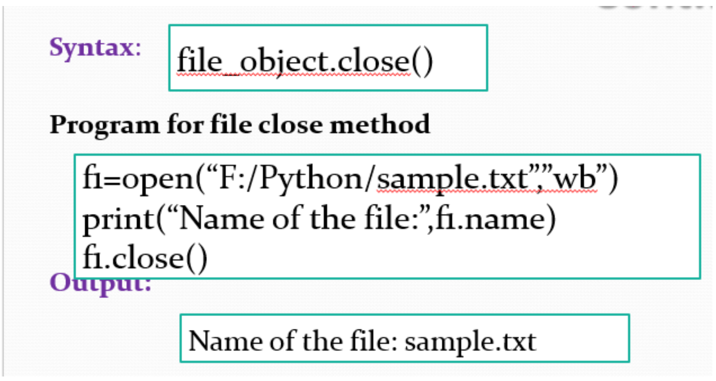
2	**rb**	Open a file for the read-only mode in the binary format.
3	**r+**	Opens a file for both reading and writing.
4	**rb+**	Open a file for read and write only mode in the binary format.
5	**w**	**Write Mode:** This mode is used when you want to write data into the file or modify it. Remember write mode overwrites the data present in the file.
6	**wb**	Open a file for write only mode in the binary format.
7	**w+**	Opens a file for both reading and writing.
8	**wb+**	Opens a file for both writing and reading in binary format.
9	**a**	**Append Mode:** Append mode is used to append data to the file. Remember data will be appended at the end of the file pointer.
10	**ab**	Open a file for appending only mode in the binary format.
11	**a+**	**Append or Read Mode:** This mode is used when we want to read data from the file or append the data into the same file.
12	**ab+**	Open a file for appending and read-only mode in the binary format.

2. CLOSING A FILE

The close() Method

To close a file object we will use close() method. The file gets closed and cannot be used again for reading and writing.

The close() method of the file object flushes any unwritten information and closes the file object, after which no more writing can be done.



Syntax:

```
file_object.close()
```

Program for file close method

```
fi=open("F:/Python/sample.txt","wb")
print("Name of the file:",fi.name)
fi.close()
```

Output:

```
Name of the file: sample.txt
```