Programming and Problem Solving through Python Language
O Level / A Level

Chapter - 7 : File Processing

Concepts of Files

- Many real-world problems handle large volume of data and in such situations external storage devices like the floppy disk and the hard disks are used.
- Data is stored in these devices using the concept of files.
- A file is a collection of related data stored on a particular area of the disk.

Filenames

- Every disk file has a name, and one must use filenames when dealing with disk files.
- Filenames are stored as strings, just like other text data.
- The rules as to what is acceptable for filenames and what is not, differ from one operating system to another.

File Opening in Various Modes and Closing of a File

Opening a File

- The process of creating a stream linked to a disk file is called opening the file.
- When one opens a file, it becomes available for
  - reading (meaning that data is input from the file to the program),
  - writing (meaning that data from the program is saved in the file), or
  - both.
- After working with the file, close the file.

Syntax File_Pointer = open(filename , mode)

Example
```
fp=open("test.txt", 'r')   open the text file for reading
fp=open("test.txt", 'rt')  open the text file for reading
```

Modes of Opening a File

Character	Mode	Description
"r"	Read	Default value. Opens a file for reading, error if the file does not exist
"a"	Append	Opens a file for appending, creates the file if it does not exist
"w"	Write	Opens a file for writing, creates the file if it does not exist
"x"	Create	Creates the specified file, returns an error if the file exists
"+"	Update	Open a file for reading and writing . Used with r+ ,a+
"t"	Text	Default value. Text mode eg. rt , wt , at, xt
"b"	Binary	Binary mode (e.g. images) eg. rb , wb, ab, xb

Closing a File

- When we are done the operations on the file, we need to close the file.
- Closing a file will free up the resources attached with the file.
- It is done using the **close**( ) method available in Python.

Syntax File_Pointer.close( )

Example

```
fp=open("test.txt", 'w')

fp.close( )
```

Writing a File

- Writing a string or sequence of bytes (for binary files) is done using the **write**( ) method.
- To write into a file in Python, it is need to open it in write(w), append(a) or exclusive creation(x) mode.
- The write(w) mode, overwrite into the file if it already exists.

Syntax File_Pointer.write(string)

Example

```
fp=open("test.txt", 'w')
fp.write("My File-1\n")
fp.write("Line-2\n")
fp.write("Line-3")
fp.close( )
```

Reading a File

- To read the content from the file use the **read**(size) method to read data.
- If the size parameter is not specified, it reads and returns up to the end of the file.
- To read from a file in Python, it is need to open it with read(r) mode.

Syntax File_Pointer.read(size)

Example

```
fp=open("test.txt", 'r')
#Read the first 4 character
x=fp.read(4)
print(x)

#Read the next 4 character
x=fp.read(4)
print(x)

#Read the remaining character
```

```
x=fp.read( )
print(x)
fp.close( )
```

Output

```
My F

ile-

1
Line-2
Line-3
```

Reading a Line from File

- We use the readline( ) method to read individual lines of a file.

Syntax File_Pointer.readline( )

Example

```
fp=open("test.txt", 'r')

#Read the first line
x=fp.readline( )
print(x)

#Read the next line
x=fp.readline( )
print(x)

#Read the next line
x=fp.readline( )
print(x)
fp.close( )
```

Output

```
My File-1

Line-2

Line-3
```

Reading a complete File line by line

- We can read a file line-by-line using a for loop

Syntax

```
for line in File_Pointer:
        x=line
```

Example

```
fp=open("test.txt", 'r')

for line in fp:
    x=line
    print(x)
fp.close( )
```

Output

```
My File-1

Line-2

Line-3
```