CHAPTER-5 FILE HANDLING

WORKING WITH BINARY FILES:

Binary files are used to store binary data such as images, video files, audio files etc. They store data in the binary format (0's and 1's) .In Binary files there is no delimiter for a line. To open files in binary mode, when specifying a mode, add 'b' to it.

Pickle module can be imported to write or read data in a binary file.

5.7 (a) Write data to a Binary File:

Example:

```
import pickle
e={'Namita':25000,'Manya':30000,'Tanu':20000}
f1=open("D:emp.dat","wb")
pickle.dump(e,f1)
f1.close()
```

Output:

A file named emp.dat will be created in current working directory.

(b) Read the data from Binary File:

Example:

```
import pickle
1=open('D:emp.dat','rb')
e=pickle.load(f1)
for x in e:
  print(x)
f1.close()
```

Output:

Namita
Manya
Tanu

Example :

```
import pickle
f1=open('emp.dat','rb')
e=pickle.load(f1)
for x in e:
 if(e[x]>=25000 and e[x]<=30000):
 print(x)
f1.close()
```

Output:

tell( ) and seek( ) methods:

- tell( ): It returns the current position of cursor in file.

Example 5.8:

```
fout=open("story.txt","w")
fout.write("Welcome Python")
 print(fout.tell( ))
fout.close( )
```

Output:

14

- seek(offset) : Change the cursor position by bytes as specified by the offset.

seek() method

In Python, `seek()` function is used to **change the position of the File Handle** to a given specific position. File handle is like a cursor, which defines from where the data has to be read or written in the file.

Syntax: *f.seek(offset, from_what), where f is file pointer*
Parameters:
Offset: *Number of postions to move forward*
from_what: *It defines point of reference.*

The reference point is selected by the **from_what** argument. It accepts three values:

- **0:** sets the reference point at the beginning of the file
- **1:** sets the reference point at the current file position
- **2:** sets the reference point at the end of the file

By default from_what argument is set to 0.

Note: Reference point at current position / end of file cannot be set in text mode except when offset is equal to 0.

SPECIAL NOTE: To read data from current location in text file also you need to put “b” with its mode for 3.X.X versions in PYTHON.

Example 5.9:

```
fout=open("story.txt","w")
fout.write("Welcome Python")
fout.seek(5)
 print(fout.tell( ))
 fout.close( )
```

Output:

5

File I/O Attributes

Attribute	Description
name	Returns the name of the file (Including path)
mode	Returns mode of the file. (r or w etc.)
encoding	Returns the encoding format of the file
closed	Returns True if the file closed else returns False

Example:

```
f = open("D:\\story.txt", "r")
print("Name of the File: ", f.name)
 print("File-Mode : ", f.mode)
print("File encoding format : ", f.encoding)
print("Is File closed? ", f.closed)
f.close()
print("Is File closed? ", f.closed)
```

OUTPUT:

```
Name of the File: D:\story.txt File-Mode : r
File encoding format : cp1252
Is File closed? False
Is File closed? True
```

Relative and Absolute Paths :

 We all know that the files are kept in directory which are also known as folders.
 Every running program has a current directory. Which is generally a default directory and python always see the default
directory first.
 The absolute paths are from the topmost level of the directory structure. The relative paths are relative to the current
working directory denoted as a dot(.) while its parent directory is denoted with two dots(..).
 OS module provides many such functions which can be used to work with files and directories. OS means Operating System.
 getcwd( ) is a very function which can be used to identify the current working directory

```
>>> import os
>>> cwd=os.getcwd()
>>> print(cwd)
C:\Users\kv2kkdSrSec\AppData\Local\Programs\Python\Python36-32
```

STANDARD FILE STREAMS :

• We use standard I/O Streams to get better performance from different I/O devices.
• Some Standard Streams in python are as follows –
– Standard input Stream sys.stdin
– Standard output Stream sys.stdout
– Standard error Stream sys.stderr

```
import sys
f=open(r"hello.txt")
line1=f.readline()
line2=f.readline()
line3=f.readline()
sys.stdout.write(line1)
sys.stdout.write(line2)
sys.stdout.write(line3)
```

```
sys.stderr.write("\nNo errors occured\n")
f.close()
```

OUTPUT:

```
Comp Science
Informatics Practices
Opjindal
No errors occured
```