FILE HANDLING in Python

Setting offsets in a file:
- seek() method
- tell() method

The pickle module
- dump() method
- load() method

Absolute and relative paths

CSV file handling

Setting offsets in a file

- The functions that we have learnt till now are used to access the data sequentially from a file.

- But if we want to access data in a random fashion, then Python gives us the following functions to do so:

  - seek() : position the file object at a particular location

  - tell() : specifies the current position of the file object

Setting offsets in a file: tell() method

- This function returns an integer that specifies the current position of the file object in the file.
- The position so specified is the byte position from the beginning of the file till the current position of the file object.
- The syntax of using tell() is:

```
file_object.tell()
```

Setting offsets in a file: seek() method

- This method is used to position the file object at a particular position in a file.

- The syntax of seek() is:

  file_object.seek(offset [, reference_point])

  where offset is the number of bytes by which the file object is to be moved.

  reference_point indicates the starting position of the file object. That is, with reference to which position, the offset has to be counted.

Setting offsets in a file: seek() method

- Reference point can have any of the following values:

  0 - beginning of the file

  1 - current position of the file

  2 - end of file

- By default, the value of reference_point is 0, i.e. the offset is counted from the beginning of the file.

  For example,

  the statement fileObject.seek(5,0) will position the file object at 5^{th} byte position from the beginning of the file.

Manipulation of data in a text file

- So far we have learnt various methods that help us to
  - open and close a file,
  - read and write data in a text file,
  - find the position of the file object and
  - move the file object at a desired location,
- let us now perform some basic operations on a text file:
  - Creating a file and writing data
  - Traversing a file and displaying data

Creating and writing data in a text file

- To create a text file, we use the open() method and provide the filename and the mode.
- If the file already exists with the same name, the open() function will behave differently depending on the mode (write or append) used.
  - If it is in write mode (w), then all the existing contents of file will be lost, and an empty file will be created with the same name.
  - But, if the file is created in append mode (a), then the new data will be written after the existing data.
- In both cases, if the file does not exist, then a new empty file will be created.

Traversing a file and displaying data

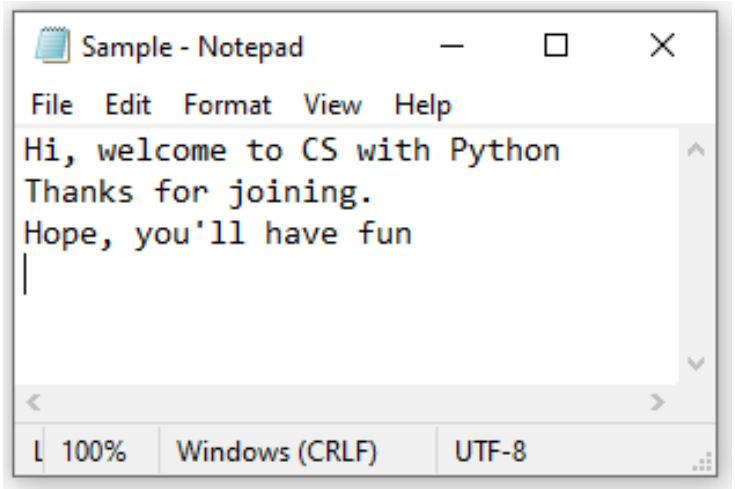
- To read and display data of a file, file needs to be opened in read mode and reading will begin from beginning of the file.

sample program

- Suppose, we have written the following program:

```
1  #witing into a text file
2  f = open("Sample.txt",'w')
3  f.write("Hi, welcome to CS with Python\n")
4  f.write("Thanks for joining.\n")
5  f.write("Hope, you'll have fun\n")
6  f.close()
```

which will result in the formation of a text file with name “Sample.txt”, and have the following content



Code:

```
#witing into a text file
f = open("Sample.txt",'w')
f.write("Hi, welcome to CS with Python\n")
f.write("Thanks for joining.\n")
f.write("Hope, you'll have fun\n")
f.close()
print("Reading the contents of file")
g = open("Sample.txt")
print(g.read())
print("Position of file object is: ", g.tell())

g.seek(0)          #will set the file position at the starting of file
print("Now, position of file object is: ", g.tell())

g.seek(5)
print("Now, position of file object is: ", g.tell())

print(g.read())
g.close()           #closing the file
```

Output:

```
In [1]: runfile('C:/Users/Vaibhav/Desktop/temp.py', wdir='C:/Users/Vaibhav/Desktop')
Reading the contents of file
Hi, welcome to CS with Python
Thanks for joining.
Hope, you'll have fun

Position of file object is:  75
Now, position of file object is:  0
Now, position of file object is:  5
elcome to CS with Python
Thanks for joining.
Hope, you'll have fun
```

The pickle module

- We know that Python considers everything as an object. So, all data types including list, tuple, dictionary, etc. are also considered as objects.
- During execution of a program, we may require to store current state of variables so that we can retrieve them later to its present state.
- Suppose you are playing a video game, and after some time, you want to close it. So, the program should be able to store the current state of the game, including current level/stage, your score, etc. as a Python object.

The pickle module

- Likewise, we may like to store a Python dictionary as an object, to be able to retrieve later. To save any object structure along with data, Python provides a module called pickle.

- The module pickle is used for serializing and de-serializing any Python object structure.

Pickling is a method of preserving food items by placing them in some solution, which increases the shelf life. In other words, it is a method to store food items for later consumption.

The pickle module

- Serialization is the process of transforming data or an object in memory (RAM) to a stream of bytes called byte streams.
- These byte streams in a binary file can then be stored in a disk or in a database or sent through a network. Serialization process is also called pickling.
- De-serialization or unpickling is the inverse of pickling process where a byte stream is converted back to Python object.

The pickle module

- The pickle module deals with binary files. Here, data are not written but dumped and similarly, data are not read but loaded.

- The pickle Module must be imported to load and dump data. The pickle module provides two methods - dump() and load() to work with binary files for pickling and unpickling, respectively.

The pickle module: dump() method

- This method is used to convert (pickling) Python objects for writing data in a binary file. The file in which data are to be dumped, needs to be opened in binary write mode (wb).

- Syntax of dump() is as follows:

  dump(data_object, file_object)

  where data_object is the object that has to be dumped to the file with the file handle named file_object

- We need to close the file after pickling (dump())

The Pickle module: dump() method

- for e.g.)

```
import pickle
L = [1,"Geetika",'F', 26]
fb = open("SampleB.dat", "wb")
pickle.dump(L,fb)
fb.close()
```

pickle module must be imported first before using the dump() method

The pickle module: load() method

- This method is used to load (unpickling) data from a binary file. The file to be loaded is opened in binary read (rb) mode.

- Syntax of load() is as follows:

  Store_object = load(file_object)

  Here, the pickled Python object is loaded from the file having a file handle named file_object and is stored in a new file handle called store_object. We need to close the file after pickling (dump())

The pickle module: load() method

- for e.g.)

code:

```
import pickle
fb = open("SampleB.dat", "rb")
print("Data in binary file is: ")
s = pickle.load(fb)
print(s)
fb.close()
```

Output:

```
In [3]: runfile('C:/Users/Vaibhav/Desktop/temp.py',
wdir='C:/Users/Vaibhav/Desktop')
Data in binary file is:
[1, 'Geetika', 'F', 26]
```

Absolute and relative paths

- An absolute path refers to the same location in a file system relative to the root directory, whereas a relative path points to a specific location in a file system relative to the current directory you are working on.
- For e.g.)

  D: \documents\mydocument.doc

  mydocument.doc

Note:

In windows, you can right-click on the file and click on properties to determine the absolute path of that file

CSV file handling

- What is CSV?

  A CSV file (Comma Separated Values file) is a type of plain text file that uses specific structuring to arrange tabular data. Because it's a plain text file, it can contain only actual text data.

- The structure of a CSV file is given away by its name. Normally, CSV files use a comma to separate each specific data value. Here's what that structure looks like:

CSV

```
column 1 name,column 2 name, column 3 name
first row data 1,first row data 2,first row data 3
second row data 1,second row data 2,second row data 3
...
```

CSV file handling

- In general, the separator character is called a delimiter, and the comma is not the only one used. Other popular delimiters include the tab (\t), colon (:) and semi-colon (;) characters

- Opening and closing of a csv file is similar to text files

- However, in order to read and write to csv files, we can use functions defined under csv module:

  - reader()
  - writerow()

Summary

- Introduction to files
- Types of files:
  - text file
  - binary file
- Opening a file
  - file open modes
- closing a file
- Opening a file using with clause

Summary

- Writing to a text file
  - write() method
  - writelines() method
- Reading from a file
  - read() method
  - readline() method
  - readlines() method
- Setting offsets in a file:
  - seek() method
  - tell() method

Summary

- Setting offsets in a file:
  - seek() method
  - tell() method
- The pickle module
  - dump() method
  - load() method
- Absolute and relative paths
- CSV file handling