FILE HANDLING Concept

- In any programming languages, program memory allocation takes place in RAM area at the time of execution of program .
- RAM is a volatile, so that after execution of the program the memory is going to be de-allocated.
- Any programming languages programs are good at processing the data but they can not store data in permanent manner
- After processing the data , we have to store the data in perminent manner.
- If the data is stored in permanent manner, we can use the data whenever we want.
- We can store the data in permanent manner by using file.
- File is a named location on disk , which is used to store the related data in permanent manner in the hard disk.
- By using a python programs, we can put the data into a file , we can get the data from a file & we can modify the data of the file.
- Before going to read or write or modify the data of the file, we have to open the file.
- By using open() , we can open the file.
- At the time of opening the file, we have to specify the file modes.
- Mode of the file indicates what purpose we are going to open the file.

Syntax: file_object = open('demo.txt' , 'r')

- here open() method taking **file_name** as first parameter and required **file_mode** as second parameter.

MODES OF FILE

r ---- > open a file for reading (default)

w -----> open a file for writing. It creates a new file if file does not exists or truncates the file if exists.

x -----> open a file for exclusive creation. If the file already exists, the operation fails.

a ----> open a file for appending data at the end of the file with out truncating , it creates a new file if file does not exist.

t ------> open a text mode(by default)

b --->open a binary mode.

syntax : x = open("fileName" , "modes of the file")

- After executing the open function, it will create a file object with the specified modes like x.
- BY calling the methods on the file object, we can read the data from the file , we can write into file or update the data of a file.
- After performing the operations on the file object we have to close the file object.
- By using close() we can close the file object .

READ DATA:

- To read the data from file object , in python we have read(), readline() and readlines() methods.
- Using empty **read()** method we can read entire file object data from where a file_pointer is available and it returns in the form of string type object.

Note: Create a file with name **demo.py** with some content

python is easy language
python is more powerfull language
python is dynamic

For Example:

Q) Write a python program to read the data from the file?

```
file_object = open("demo.txt" , "r" )
print ("file object is opened")
print(file_object)
data = file_object.read()
print(data)
```

read(count_number)

read() will take the 'int' value as input and It reads given integer number of characters of the file and return as a string format.

```
file_object = open("demo.txt")
print ("file is opened")
print(file_object)
data = file_object.read(10)
print(data)
```

Note: here first 10 charecters reading from the given file data.

Note: If specified **file is not available** then we will get exception like bellow,

FileNotFoundError: [Errno 2] No such file or directory: 'demo.txt'

readline():

- Python facilitates to read the file line by line by using a function readline() method.
- The empty readline() method reads the current line of the file from the file pointer location, i.e., if we use the readline() method two times, then we can get the first two lines of the file.
- It is used to Read the entire single line of file where our file pointer is available.
- Consider the following example which contains a function readline() that reads the first line of our file "demo.txt" containing three lines.

For example:

Q) Write a python program to read the first line data from the file ?

```
file_object = open("demo.txt")
print ("file is opened")
print(file_object)
data = file_object.readline()
print(data)
```

readline(count_number)

- The readline(int) will reads the given specified number of charecters only from the file pointer location but not upto end of the line.
- If given integer value is more than given file data length then readline(int) will read the data which is available length in a file.

For example:

```
fileObject = open('demo.txt', 'r')
data = fileObject.readline(2)
print(data)    # 'py'
```

readlines()

- Is used to Read the entire all lines of file where our file pointer is available.
- It returns output as a list of items, where each line is a new string of list format.

For example:

```
fileObject = open('demo.txt', 'r')
data = fileObject.readlines()  # ['line1' , 'line2' , 'line3']
print(data)   #
```

Output : ['python is easy language\n' , 'python is more powerfull language\n' , 'python is dynamic']

tell()

- Python provides the tell() method which is used to print the byte number at which the file pointer currently exists.

```
# open the file demo.txt in read mode
fileptr = open( "demo.txt" , "r")
#initially the filepointer is at 0
print("The filepointer is at byte :", fileptr.tell() )
#reading the content of the file
content = fileptr.read();
#after the read operation file pointer modifies. tell() returns the location of the fileptr.
```

```
print("After reading, the filepointer is at:",fileptr.tell())
```

Output:

```
The filepointer is at byte : 0
After reading, the filepointer is at: 55
```

Modifying file pointer position

seek()

- In real-world applications, sometimes we need to change the file pointer location externally since we may need to read or write the content at various locations.
- For this purpose, the Python provides us the seek() method which enables us to modify the file pointer position externally.
- The syntax to use the seek() method is given below.

Syntax: <file_object>.seek(offset, from)

The **seek()** method accepts two parameters:

offset: It refers to the new position of the file pointer within the file.

from: It indicates the reference position from where the bytes are to be moved.

--->> If it is set to 0, the beginning of the file is used as the reference position.

--->> If it is set to 1, the current position of the file pointer is used as the reference position.

--->> If it is set to 2, the end of the file pointer is used as the reference position.

Note:

- using seek(), We can change the file pointer position from one location to another location.
- By default file pointer points the zero location when ever open the file.

Example:

```
# open the file demo.txt in read mode
fileptr = open("demo.txt","r")
# initially the filepointer is at 0
print("The filepointer is at byte :",fileptr.tell())
#changing the file pointer location to 10.
fileptr.seek(10);
# tell() returns the location of the fileptr.
print("After reading, the filepointer is at:",fileptr.tell())
```

Output:

```
The filepointer is at byte : 0
After reading, the filepointer is at: 10
```

Practice Examples:

Q. How to find total available lines from a given file demo.txt ?

```
file_object = open('demo.txt','r')
data = file_object.readlines()
print(len(data))
print(data)
```

Output:

```
available lines are : 3
['Python is easy language\n', 'Python is powerfull\n', 'Python is open source\n']
```

Q. How to finding the total available charecters in a given file ?

```
file_object = open('demo.txt','r')
data = file_object.read()
print('Total avaliable charecters in a given file is :',len(data))
```

Output: Total avaliable charecters in a given file is : 66

Q. How to finding total number of words from a given file and return all words?

```
file_object = open('demo.txt','r')
data = file_object.read()
list_of_words = data.split()
print('Total number of words in a given file is :',len(list_of_words))
print('Available words are :',list_of_words)
```

Output:

```
Total number of words in a given file is : 11
Available words are : ['Python', 'is', 'easy', 'language', 'Python', 'is', 'powerfull',
'Python', 'is', 'open', 'source']
```

Q. How to display every word count how many times comming from a given file with word is a key and word count is a value format ?

```
file_object = open('demo.txt','r')
data = file_object.read()
list_of_words = data.split()
d = {}
for word in list_of_words:
    if word not in d:
        d[word] = 1
    else:
        d[word] = d[word]+1
print(d)
```

Output:

```
{'Python': 3, 'is': 3, 'easy': 1, 'language': 1, 'powerfull': 1, 'open': 1, 'source': 1}
```

Q. How to find the file pointer location where available and how to change file pointer location from one place to another place ?

```
file_object = open('demo.txt','r')
data = file_object.tell()
print('file pointer available in ',data,'position')
data2 = file_object.readline(4)
print(data2)
data = file_object.tell()
print('file pointer available in ',data,'position')
file_object.seek(0)
data = file_object.tell()
print('file pointer available in ',data,'position')
```

Output:

```
file pointer available in  0 position
Pyth
file pointer available in  4 position
file pointer available in  0 position
```