File and I/O handling

File and I/O handling

- To perform any operation on the file , firstly they need to be open.
- Once the file is opened after performing certain operations it needs to be closed.

Opening and Closing Files

- **Open Function**:- To perform any read and write operation on the files it should be first opened.

Syntax:-

```
file_obj=open(file_name [,access_mode][,buffering])
```

- **file_name :** Name of the file you need to access.
- **access_mode:** It is the mode in which the file has to be opened,i.e., read, write , append etc.
- **buffering:** If buffering is 0 no buffering takes place. If value is 1 line buffering action is performed with the indicated buffer size.

Access Modes

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file.It is the default mode.
rb	Opens file to read only in binary format. File pointer is placed at the beginning.
r+	Opens a file for both reading and writing.
rb+	Opens a file for both reading and writing in binary format
w	Opens a file for writing only
wb	Opens a file for writing only in binary format
w+	Opens a file for both writing and reading
wb+	Opens a file for both writing and reading in binary format.
a	Opens a file for appending. If the file exists the pointer is at the end of the file.
ab	Opens a file for appending in binary format.
a+	Opens a file for appending and reading
ab+	Opens a file for appending and reading in the binary format.

File Object Attributes

To get various information from the file various object attributes are used which are: -

Attribute	Description
file.closed	Returns true if file is closed, false otherwise.
file.mode	Returns access mode with which file was opened.
file.name	Returns name of the file

close() and with()method

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

Syntax:-

```
file_name.close()
```

- **with() method**

If you think having to put close() every time you're done with a Python file is bunk, use the 'with' statement.

```
with open('To do.txt') as f:
    f.read()
```

Reading and Writing Files

- **write() method**: -It writes any string to an open file. The write() method does not add a newline character("\n") to the end of the string:-

```
Syntax: -
```

```
fileObject.write(string)
```

- **read() method**: -It reads a string from an open file.

```
Syntax: -
```

```
fileObject.read([count])
```

- Count reads the number of bytes from the opened file. If value of the count is given it will read until that value else will read the entire file.

Reading and Writing Files

- **file_name.**`read()` – file's entire contents as a string
- **file_name.**`readline()` – next line from file as a string
- **file_name.**`readlines()` – file's contents as a list of lines

The lines from a file object can also be read using a `for` loop

`Syntax:-`

```
name = open("filename")
    for line in name:
        statements
```

- **file_name.**`write(`**str**`)` – writes the given string to the file

Task 1

Q1. Write a function `input_stats` that accepts a file name as a parameter and that reports the longest line in the file.

For example input file, `carroll.txt`:

```
Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch.
```

```
input_stats("carroll.txt")
longest line = 42 characters
the jaws that bite, the claws that catch,
```

Task 2

- Suppose we have this `hours.txt` data:

```
123 Suzy 9.5 8.1 7.6 3.1 3.2
456 Brad 7.0 9.6 6.5 4.9 8.8
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

- Compute each worker's total hours and hours/day.
  - Assume each worker works exactly five days.

```
Suzy ID 123 worked 31.4 hours: 6.3 / day
Brad ID 456 worked 36.8 hours: 7.36 / day
Jenn ID 789 worked 39.5 hours: 7.9 / day
```

Task 3

- Write code to read a file of gas prices in USA and Belgium:

```
8.20    3.81    3/21/11
8.08    3.84    3/28/11
8.38    3.92    4/4/11
…
```

- Output the average gas price for each country to an output file named `gasout.txt`.

File Positions

- **tell()** method: -It tells the current position within the file.

- **seek(offset[,form]**: - It will change the current file position.

If form is set to 0 the beginning of the file is used as the reference position. If it is set to 1 then the current position is used as the reference position. If it is set to 2 then the end of the file would be taken as the reference position.

Renaming and Deleting Files

- Python os module provides methods that help you perform file-processing operations, such as renaming and deleting files.

- To use this module, you need to import it first and then you can call any related functions.

rename() and remove()method

- **rename() method :** -takes two arguments, the current filename and the new filename.

```
Syntax: -
 os.rename(current_file_name, new_file_name)
```

- **remove() method:** - It is use to delete files by supplying the name of the file to be deleted as the argument.

```
Syntax:-
                os.remove(file_name)
```

Directories in python

- All files are contained within various directories, and Python has no problem handling these too.

- The **os** module has several methods that help you create, remove, and change directories.

Methods in Directories

➢mkdir() method

- You can use the mkdir() method of the **os** module to create directories in the current directory.
- You need to supply an argument to this method, which contains the name of the directory to be created.

Syntax: -

```
os.mkdir("newdir")
```

➢chdir()method

- You can use the chdir() method to change the current directory.
- The chdir() method takes an argument, which is the name of the directory that you want to make the current directory.

Syntax: -

```
os.chdir(" Directory_name")
```

Methods in Directories

➢rmdir() method

- It deletes the directory, which is passed as an argument in the method.
- Before removing a directory, all the contents in it should be removed.

Syntax: -

```
os.rmdir('dirname')
```

➢getcwd() method

It is use to get the current working directory.

Syntax: -

```
os.getcwd()
```