NOTE : The write methods returns none

File handling

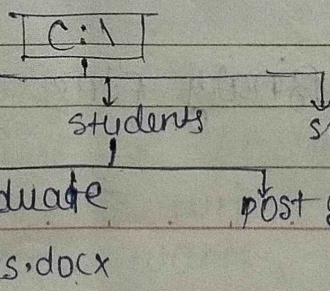
- A file is a collection of data stored on a secondary storage device like hard disk.
- Python supports file handling and allows users to handle files i.e. to read and write files, along with many other file handling options, to operate on files etc.
- working with files can be refer as working with notebooks
- To use a notebook, you must first open it. Once the notebook is opened, you can read the contents that you had previously written in it or write some new content into it. After using the notebook, you close it.
- The same concept can be applied to files. we first open a file, read or write to it, and then finally close it.

File path

- files that we use are stored on a storage medium like the hard disk in such a way that they can be easily retrieved as and when required.
- Most file systems that are used today stores files in a tree (or hierarchical) structure.
- at the top it consist root nodes. root nodes may be one or many. under the root node, there are other files and folders and each folders may contain files and folders. that means it contains limitless depth.
- File path is also known as path name

NOTE :- Folder names and file names are case sensitive in windows but they are case insensitive in linux

eg:-



while writing the file path:
- the characters after the dot (.) form the extension of the File. eg:- demo.txt here txt represents the text File type.

- The characters used to separate the Folders name is (\) slash. It is also called as delimiter

NOTE:

In windows ~~for~~ backward slash (\) and in solaris as forward slash (/) is used to delimit

Types of file path / Path name

1) ~~Relati~~ Absolute path:
- It contains the root and the complete directory list to specify the exact location of the file.
eg:- C:\students\underGraduate\BTech-CS.docx is the absolute path as all the information needed to locate the file is contained in the path

2) Relative Path:
- It needs to be combined with another path in order to access a file. ie. relative pathnames starts with respect to the current working directory and therefore lacks the leading slashes
e.g:- underGraduate\BTech_CS.docx

Types Of files

1) ASCII text files 2) Binary files

1) ASCII text Files
- A text file is a stream of characters that can be sequentially processed by a computer in forward direction.
- For this reason a text file is usually opened for only one kind of operation (reading, writing or appending) at any given time.
- Because text files can be process characters, they can only read or write data one character at a time.
- In Python, a text stream is treated as a special kind of file.
- In a text file, each line consist zero or more characters and ends with one or more characters that specify the end of line.
- each line in a text file can have maximum 255 characters
- In a text file each line of data ends with a newline character. Each file ends with a character called the end-of-file (EOF) marker

2) Binary Files
- A binary file is a file which may contain any type of data, encoded in binary form for computer storage and processing purposes.
- It includes files such as word processing documents, PDFs, images, spreadsheets, videos, zip files, and other executable program.
- like a text file, a binary file is a collection of bytes.
- It is also referred as a character stream with following two essential differences
  1) Binary file does not require any special processing of data and each data is transferred to or from the disk unprocessed
  2) python places no constructs on the file, and it may be read from, or written to, in any ~~manner~~ manner the programmer wants

- Binary Files can be processed sequencially or randomly
- Binary files store data in the internal representation format.
- It also ends with an EOF marker
- In a text file, an integer value 123 will be stored as a sequence of 3 characters - 1, 2, 3. As each ch. takes 1 byte, therefore, to store 123 it requires 3 bytes. But in Binary files same value 123 will be stored in 2 bytes in the binary form.
- This clearly indicates that binary files takes less space than text files
- Binary files can't read by text editors like Notepad.
- If you open a binary file in Notepad, you will see some scrambles, and absurd data.
- Binary files are mainly used to store data beyond text such as images, executables, etc.

21/2/22

Sem 2

File handeling

- When a program is being executed its data is stored in RAM which makes accesses faster by the CPU.
- It is consider as volatile which means when the program ends or the computer is shut down the data is lost
- Data on non-volatile storage media is stored in named location

file path

2 Types of file path

1) absolute path (from root)
2) relative path (part of absolute path)

Types of files

1) Text file
2) Binary file

Opening and closing of files

1) open function

To open a file before reading and writing to it use a built in function open() it creates a file object

Syntax:

```
Fileobj = open(filename,[,access_mode])
```

```
file = open("fycs.txt","r")
print(file)
```

Camlin Page
Date / /

Note :- access mode is an optional parameter and the default access mode is read

1) "r" - It is a default mode for opening a file for only reading a file. The file pointer is placed at the beginning of the file

2) "rb" - Reading only in binary format.

3) "r+" - Opens for both reading and writting

4) "rb+" - Reading, writing in binary format

5) "w" - file for writing only i) If the file doesn't exist it creates a file for writing
ii) If the file already exist the contents are over written / appended.

6) "wb" - writing only binary format

7) "w+" - Reading, writing

8) "wb+" - Reading, writing in binary format.

9) "a" - Opens a file for appending the file pointer is placed at the last of the file if the file exist. If the file doesn't exist it will create new file for writing

10) "ab" - appending binary format of file

11) "a+" - both for reading and writing

12) "ab+" - reading writing in binary format

File object attributes

object name

1) file.name ← (It prints the file name)
2) file.closed ← (T or F)
3) file.mode ← it prints accessmode.

2) close() :

- It is used to close the file object

Object name.close()

3/12/22

* Reading and writing files :

Write & writelines method
(for paragraph)

write()

- It is used to write a string to an already opened file. the string may include no. special ch or other symbols
- The write method does not add a newline ch (\n)

syntax :

```
fileobject.write(string);
```

① Open file with write mode ② use write function to add string close the object print the object closed

```
File = open("demo.txt", "w")
print(File)
file.write("Hello world")
# file.close()
print("Object is closed", file)
```

writelines()

It is used to write a list of strings.

```
File = open ("hello.txt","w")
lines = ["python", "Java", "C++"]
File.writelines(lines)
print (File)
```

append()

To append a file you must be using a or ab mode depending on the file type.
If you open a file in 'w' or "wb" mode then its existing contents will be overwritten
so always open the file in a or ab mode to add more data to the existing file

```
File = Open("demo.txt", "a")
lines = ["python", "Java", "C++"]
file.writelines (lines)
print (file)
```

} appending.

Read() readlines() methods

Read()

It is used to read a string from an already opened file.

Fileobject.read([count])

- Count is an optional parameter which specifies the no. of bytes to be read from the opened file
- The method start reading from beginning of the file. If count is missing or it has -ve value it reads the entire content of the file.

```
File = open("demo.txt", "r")
File.read ~~(f[0])~~ ((10))
~~print~~ (
File.close()
print ("the file is closed", file)
```

NOTE: read method returns newline as '\n"
~~readline()~~ is

readline() is used to read the single of a file
- It returns an empty string when the end of file has been reached.
- The blank line is represented by \n and readline method returns a string containing only a single new character

~~5/12/22~~

Opening a file using with keyword

The advantage of using the with keyword is the file is properly closed after it's used even if an error occurs during read or write operation or even when you forget to explicitly close the file

Write a program to print content from the file with the help of with keyword.

```
with open("File.txt", "rb") as File:
    for line in File:
        print(line)
print("let's check if the file is closed:", File.close())
```

Renaming and deleting a File

1) rename()
- It takes 2 argument current Filename and the new Filename
- The Os module in python has various methods to perform file processing operation

Syntax

```
Os.rename("old file name","new file name")
```

eg:-
```
import Os
Os.rename("File.txt","Fycs.txt")
print("renamed")
```

2) remove()
- This method is used to delete the file
- It takes filename as argument

Syntax
```
Os.remove("filename")
```

```
import Os
Os.remove("fycs.txt")
print("Fycs is deleted")
```

To make a directory
- The mkdir() of Os module is used to create directories in the current directory (i.e drives)
- It takes name of the director as argument

Syntax : (folder name)
```
Os.mkdir("Fycs")
```

getcwd()
It is used to display current working directory

Syntax:
```
Os.getcwd()
```

chdir()
It is used to change the current directory
the method takes the ~~mat~~ name of directory which you want to make the current directory as an argument

Syntax:
```
Os.chdir("Foldername")
```

rmdir()
It is used to remove or delete the directory

Syntax:
```
Os.rmdir("Foldername")
```

split() (converts into list)
python allows you to read line from a file and split the line based on the characters (treated as a string)

```
with open("File.txt","r") as file:
    line = file.readline()
    words = line.split()
    print(words)
```

file processing operation ——→ OS module

Write a program to accept a filename as the input from the user open the file and count the no. of times a character appears in a file assuming that file is already created.

```
filename = input("enter the filename:")
with open(filename) as file:
    text = file.read()
    letter = input("enter the character to be searched:")
    count = 0
    for char in text:
        if char == letter:
            count += 1
print(count)
```

7/12/22

seek(), tell() ⟸ methods

the seek() method sets the current position in the file stream

Syntax:

file_obj.seek() / file_obj.seek(offset)

Offset : It is required no. representing the position to set the current file streaming position (change the current position)

```
f.open("hello.txt", "r")
f.seek(4)
```

The

while creating class the

Date / /

seek() with negative offset

It only works when the file is open in binary mode.

tell()

It returns the current file position in the file stream.

Syntax:

file_obj.tell()

Zipping & unzipping of a file: By using zipfile module

The module zipfile creates a zip file object using the ZipFile class

- for the extract() method on the zip file object and pass the name of the file to be extracted and the path where the file needed to extracted
- zip file is a file format used for compressing multiple files together in a single file
- The zip file class contains 2 method extract() and extractall()

for unzip.

extractall():

It is used to extract all the files present in the zip file to the current working directory

Syntax:

zipfile.extractall(file_path, members=none, pwd=none)

file path :- location where the file needs to be extracted

members :- specifies the list of file to be extracted

pwd :- used for encrypted file

Camlin Page
Date / /

d) extract():
It is used to extract the member (File) from the zip to the current working directory

Syntax:
ZipFile.extract(file_path, members=none, pwd=none)

* ZipFiles takes any object to be again zip. to compress an individual file we use the method write()

- To compress an individual files into a zip
1) create a new zipfile object
2) add the files you want to compress with write()
3) Specify the path of a newly created zip as the 1st parameter
4) set the 2nd parameter mode to write