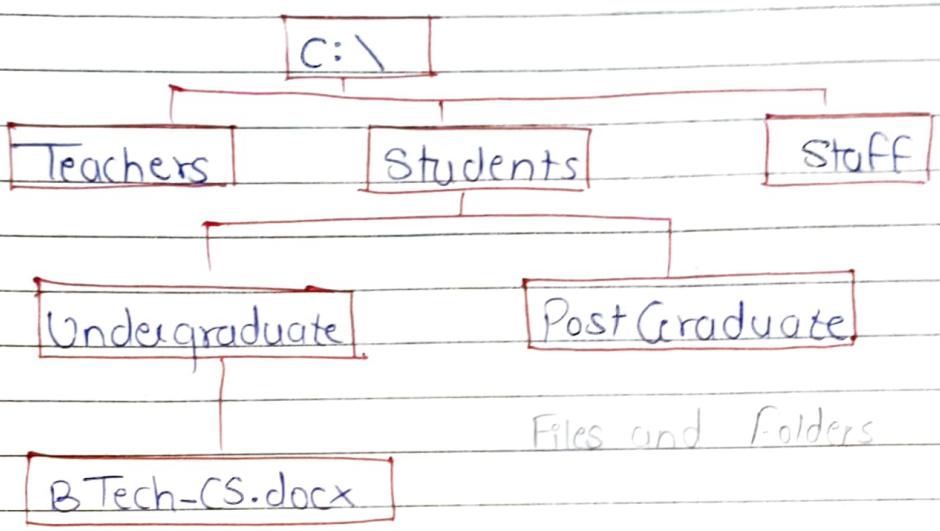
Date : / /
Page

Unit VI File Handling

* File Path :- Files are stored in a Hard Disk in such a way that they can be retrieved as when required.



- Character after the dot from the extension of the file. example: (.docx) indicates that the file is a Word document.
- character used to seperate the folder names (also called the delimiter) is specific.
- OS uses the forward slash (/)
- Microsoft Windows uses the backward slash (\)

* ~~Relative Path~~ * Absolute Path :-

C:\Students\Under Graduate\BTech-CS.docx

- A file path can be either relative or absolute
- Absolute path always contains the root and the complete dictionary list to specify the exact location the file.

Date : / /
Page

• Relative Path :- needs to be combined with another path in order to access file
• relative pathnames starts with respect to the current working dictionary
• lacks leading slashes

Student/Undergraduate/BTech/~~docx~~ - CS.docx

• If you use a relative file path from the wrong dictionary, then either the wrong file will be accessed or no file will be accessed if no file of the specified name exists in the given path

① ASCII Text Files :-

• A text file is a stream of characters, can be sequentially processed by a computer in forward direction.
• usually opend for only one kind of operation at any given time.
• 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 contains zero or more characters and ends with one or more characters ~~FE~~ that specify the end of line.
• can have maximum of 255 characters.
• integer value will be represented as a number that occupies 2 or 4 bytes of memory internally

but

Externally the integer value will be represented as a string of characters representing its decimal and hexadecimal value.

int ⇒ 123 .... 2 bytes
str ⇒ "123" .... each no. a byte 3 bytes

• each file ends with special character end-of-line (EOF)

* Binary Files :- contain any type of data, encoded in binary form for computer storage and processing purposes.
- includes files, documents, PDF's, images, videos
- Like a text file, a binary file is collection of bytes
- does not require any special processing of the data
- if file is used to store records of students, then to update a particular record, 1st locate record, put record into memory, update it, finally write record back to disk at its appropriate location in the file.
- Stores data in internal representation format.
- ends with EOF
- binary file takes less space to store same data than text file

Text files	Binary Files
① Sequential files	① contain arbitary binary data
② only stores texts	② used to store binary data such as image etc.
③ delimiter EOL (End of Line \n)	③ no delimiter
④ more time for process	④ fast time for process
⑤ Easy to understand	⑤ difficult to understand
⑥ Has extension .txt	⑥ Has extension .dat
⑦ Programming on text files is very easy	⑦ Programming on binary files is very hard.
⑧ less chances to get corrupt	⑧ easily get corrupted

Date : / /
Page

* Opening and Closing files

Syntax :

f= open (file name, mode)

~~File~~ f.close ()

example :

file = open ('geek.txt', 'r')

print (each)

* Opening File with "with" keyword

- Advantage of "with" keyword is that.
  we don't have to worry about closing the file.
- it automatically closes an opened file.
- avoids the errors that can occur due to un proper closing of files by the programmer.

~~File = open ("ABC.txt", "r")~~

~~for line in file :~~

with open ("ABC.txt","r") as file

for line in file :

print (line)

print (file.closed)

O/P => True

Date : / /
Page

* Access Modes
- r: open file for read operation
- rb: open file for reading only in binary format.
- r+: opens for reading and writing
- rb+: opens for reading and writing in binary format.
- w: open for write operation
  if file not present then it creates the file as well.
- wb: opens for writing only.
- w+: open for writing and readid,
  if file not present then it creates the file
- a: for append operation.
- a+: append and read from file
- ab: open file in binary format reading & appending
- Ab+: open file in binary format for reading, appending
- w+: same as r+ but if file is absent then it creates i.

* File Object attributes

1) Fileobj.closed

```
File = open("ABC.txt", "r")
print (File.closed)
```
O/P → False

Returns True if the file is closed and false otherwise

2) Fileobj.mode

```
File = open ("ABC.txt", "r")
print (file.mode)
```
O/P → r

Returns access mode with which file has opened

3) Fileobj.name

```
File = open ("ABC.txt", "r")
print (file.name)
```
O/P → ABC.txt

Returns name of the file.

Ex:-

```
File = open ("File1.txt", "wb")
print ("Name of file:", file.name)
print ("File is colsed", file.closed)
```

O/P Name of the file: File1.txt
File is closed. false

Date : / /
Page

* Reading and Writing files

Syntax:

- file = open ("Employees.txt", "w")

```
for i in range (3):
    name= input ("Enter the name of employee:")
    file.write (name)
    file.write ("\n")

file.close()
print ("Data is written into the file.")
```

⇒ can add n no. of lines

* Writeline()

```
f = open ("demofile 3. txt", "a")
f.writelines (["see you soon!", "Over and out."])
f.close()

f=open ("demofile3.txt","r")
print (f.read())
```

* Append () method in file

```
file1 = open ("myfile.txt", "a")
file1.write ("Today \n")
file1.close()
```

Date : / /
Page

```
Readline()
file=open("file1.txt","r")
print(file.read(10))
file.close()
```

→ count is optional
spaces are also counted
only till counts are printed

- Readline = used to read single line.
- Readlines () method is used to read all the file.

```
file=open("file1.txt","r")
print(file.readlines())
file.close()
```

⇒ can add only 1 line

- ① Write :- syntax:-

```
fileobj.write(string)
```

example:-

```
file=open("ABC.txt","w")
file.write("Welcome")
print("string is added to the fileobj.write(string)
```

- Readline

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

→ reads only 1st line

Date : / /
Page

Directory
~~Dictionary~~ Methods

mkdir() method
- can use this method of os module to create directories in the current ~~dictionary~~ directory

Syntax :-
```
os.mkdir("newdir")
```

ex :-
```
import os
os.mkdir("test.txt")
```

chdir() Method
- use to change the current directory.

Syntax :-
```
os.chdir("newdir")
```

ex :-
```
import os
os.chdir("/home/newdir")
```

getcwd() Method
- displays current working ~~dictionary~~ directory

Syntax `os.getcwd()`

ex :-
```
import os
os.getcwd()
```

os.path.join

Date : / /
Page

* rmdir () Method
  · deletes the directory
  syntax :-
  os.rmdir('dirname')

ex :- import os
os.rmdir ("/tmp/test")

* file.tell() ⇒ tells that where is your file pointer.

* seek (offset, [from])
bytes
seek (3, 1)

0 1 5
Hello everyone
Welcome all

file.read(5)
file.tell()

seek (3, 1) → from
→ after 3 bytes

* import os.rename ("old file", "new file")
example :-
import os.rename ("ABC.txt", "Aryan.txt")

* remove method
import os.remove ("Aryan.txt")