FILE HANDLING

- ✓ INTRODUCTION
- ✓ DATA FILES
- ✓ OPENING AND CLOSING FILES
- ✓ READING AND WRITING FILES
- ✓ STANDARD INPUT, OUTPUT AND ERROR STREAMS

Introduction

- **FILE HANDLING** is a mechanism by which we can read data of disk files in python program or write back data from python program to disk files.
- So far in our python program the standard input in coming from keyboard an output is going to monitor i.e. no where data is stored permanent and entered data is present as long as program is running **BUT** file handling allows us to store data entered through python program permanently in disk file and later on we can read back the data

DATA FILES

- It contains data pertaining to a specific application, for later use. The data files can be stored in two ways –
- Text File
- Binary File

Text File

- Text file stores information in **ASCII OR UNICODE** character. In text file everything will be stored as a character for example if data is “computer” then it will take 8 bytes and if the data is floating value like 11237.9876 it will take 10 bytes.
- In text file each like is terminated by special character called **EOL**. In text file some translation takes place when this EOL character is read or written. In python EOL is ‘\n’ or ‘\r’ or combination of both

Binary files

- It stores the information in the same format as in the memory i.e. data is stored according to its data type so no translation occurs.
- In binary file there is no delimiter for a new line
- Binary files are faster and easier for a program to read and write than text files.
- Data in binary files cannot be directly read, it can be read only through python program for the same.

Steps in Data File Handling

1. **OPENING FILE**
   - We should first open the file for read or write by specifying the name of file and mode.
2. **PERFORMING READ/WRITE**
   - Once the file is opened now we can either read or write for which file is opened using various functions available
3. **CLOSING FILE**
   - After performing operation we must close the file and release the file for other application to use it,

Opening File

- File can be opened for either – read, write, append.

SYNTAX:

file_object = open(filename)

Or

file_object = open(filename,mode)

** default mode is “read”

Opening File

myfile = open("story.txt")

here disk file "**story.txt**" is loaded in memory and its reference is linked to "**myfile**" object, now python program will access "**story.txt**" through "**myfile**" object.

here "**story.txt**" is present in the same folder where .py file is stored otherwise if disk file to work is in another folder we have to give full path.

Opening File

myfile = open("article.txt","r")

here "r" is for read (although it is by default, other options are "w" for write, "a" for append)

myfile = open("d:\\mydata\\poem.txt","r")

here we are accessing "poem.txt" file stored in separate location i.e. d:\mydata folder.

at the time of giving path of file we must use double backslash(\\) in place of single backslash because in python single slash is used for escape character and it may cause problem like if the folder name is "nitin" and we provide path as d:\nitin\poem.txt then in \nitin "\n" will become escape character for new line, SO ALWAYS USE DOUBLE BACKSLASH IN PATH

Opening File

myfile = open("d:\\mydata\\poem.txt","r")

another solution of double backslash is using "r" before the path making the string as raw string i.e. no special meaning attached to any character as:

myfile = open(r"d:\mydata\poem.txt","r")

File Handle

myfile = open(r"d:\mydata\poem.txt","r")

In the above example "myfile" is the file object or file handle or file pointer holding the reference of disk file. In python we will access and manipulate the disk file through this file handle only.

File Access Mode

Text File Mode	Binary File Mode	Description	Notes
'r'	'rb'	Read only	File must exists, otherwise Python raises I/O errors
'w'	'wb'	Write only	If file not exists, file is created If file exists, python will truncate existing data and overwrite the file.
'a'	'ab'	Append	File is in write mode only, new data will be added to the end of existing data i.e. no overwriting. If file not exists it is created
'r+'	'r+b' or 'rb+'	Read and write	File must exists otherwise error is raised Both reading and writing can take place
w+	'w+b' or 'wb+'	Write and read	File is created if not exists, if exists data will be truncated, both read and write allowed
'a+'	'a+b' or 'ab+',	Write and read	Same as above but previous content will be retained and both read and write.

Closing file

- As reference of disk file is stored in file handle so to close we must call the close() function through the file handle and release the file.

```
myfile.close()
```

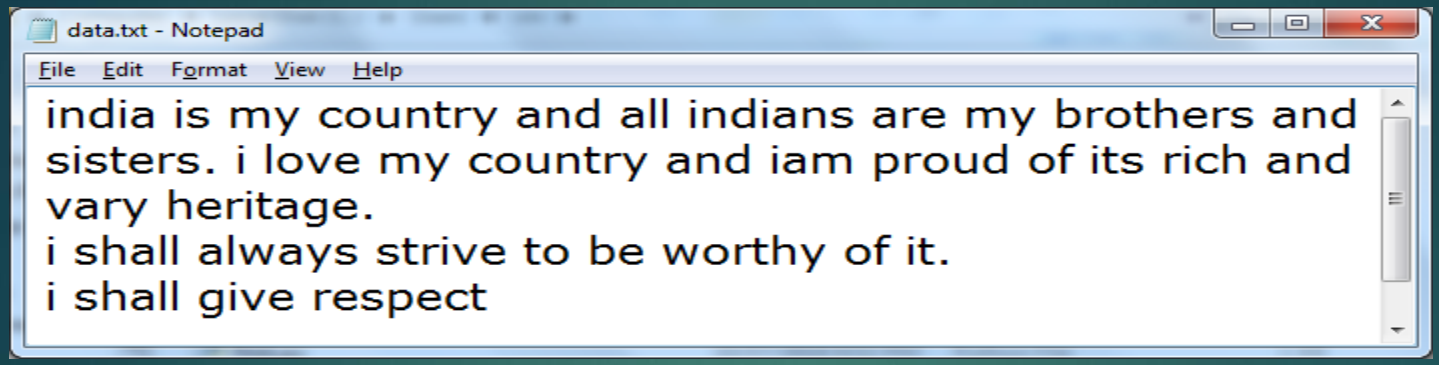
Note: open function is built-in function used standalone while close() must be called through file handle

Reading from File

- To read from file python provide many functions like :
- **Filehandle.read([n]) :** reads and return n bytes, if n is not specified it reads entire file.
- **Filehandle.readline([n]) :** reads a line of input. If n is specified reads at most n bytes. Read bytes in the form of string ending with line character or blank string if no more bytes are left for reading.
- **Filehandle.readlines():** reads all lines and returns them in a list

Example-1: read()

SAMPLE FILE

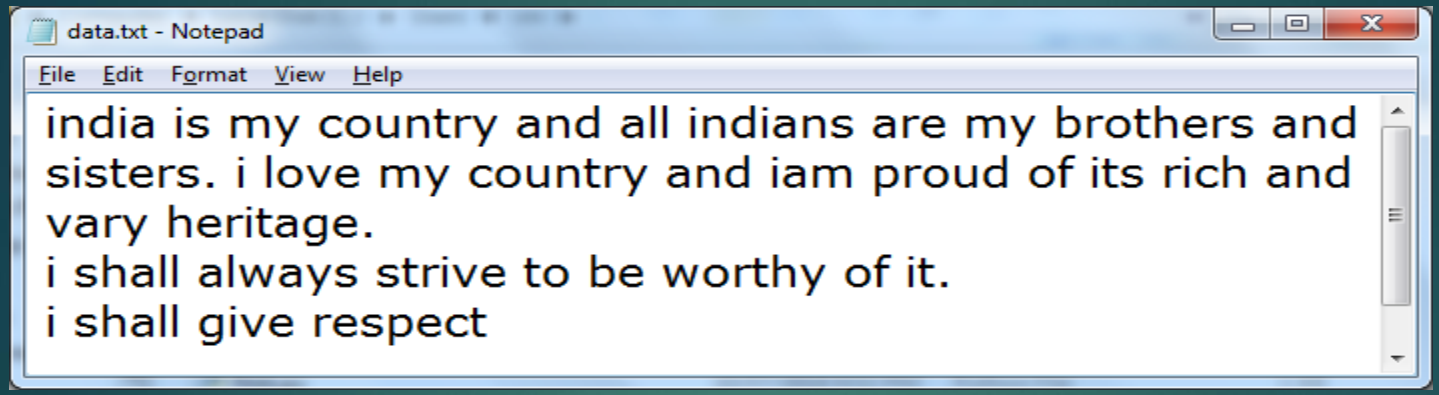



```
# data file handling program to read
myfile = open("data.txt","r")
str1 = myfile.read(10)
str2 = myfile.read(5)
str3 = myfile.read()
print("Output of First read :")
print(str1)
print("Output of Second read :")
print(str2)
print("Output of Third read :")
print(str3)
```

```
Output of First read :
india is m
Output of Second read :
y cou
Output of Third read :
ntry and all indians are my brothers and sisters. i love my country and iam prou
d of its rich and vary heritage.
i shall always strive to be worthy of it.
i shall give respect
```

Example-2: read()

SAMPLE FILE




```
# data file handling program to read
myfile = open("data.txt","r")
str1 = myfile.read()
str2 = myfile.read(10)
str3 = myfile.read(5)
print("Output of First read :")
print(str1)
print("Output of Second read :")
print(str2)
print("Output of Third read :")
print(str3)
```

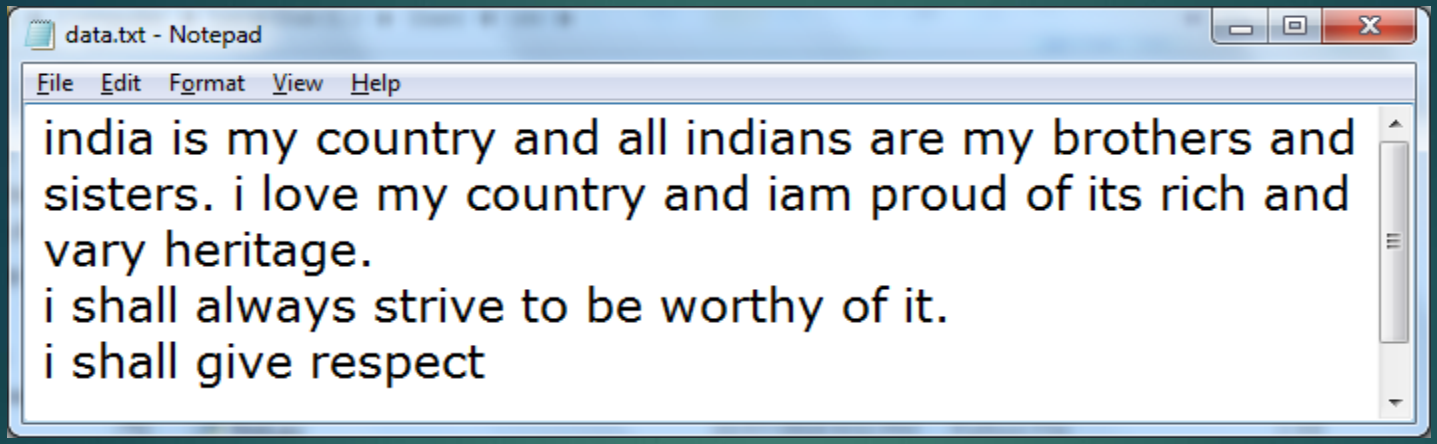
```
Output of First read :
india is my country and all indians are my brothers and sisters. i love my count
ry and iam proud of its rich and vary heritage.
i shall always strive to be worthy of it.
i shall give respect

Output of Second read :

Output of Third read :
```

Example-3: readline()

SAMPLE FILE




```
# program to read line
myfile = open("data.txt","r")
line1 = myfile.readline()
line2 = myfile.readline()
line3 = myfile.readline(10)
print(line1)
print(line2)
print(line3)
```

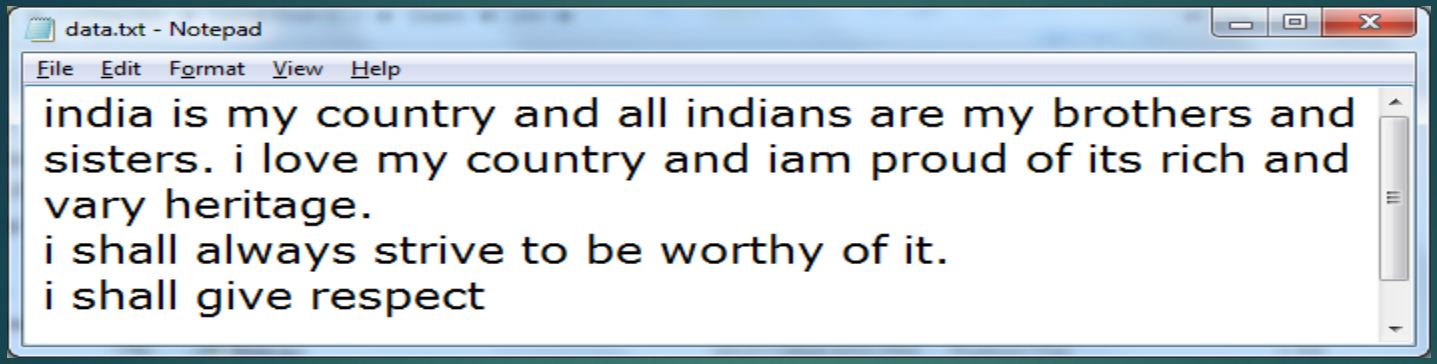
```
india is my country and all indians are my brothers and sisters. i love my count
ry and iam proud of its rich and vary heritage.

i shall always strive to be worthy of it.

i shall gi
```

Example-3: readline()

SAMPLE FILE



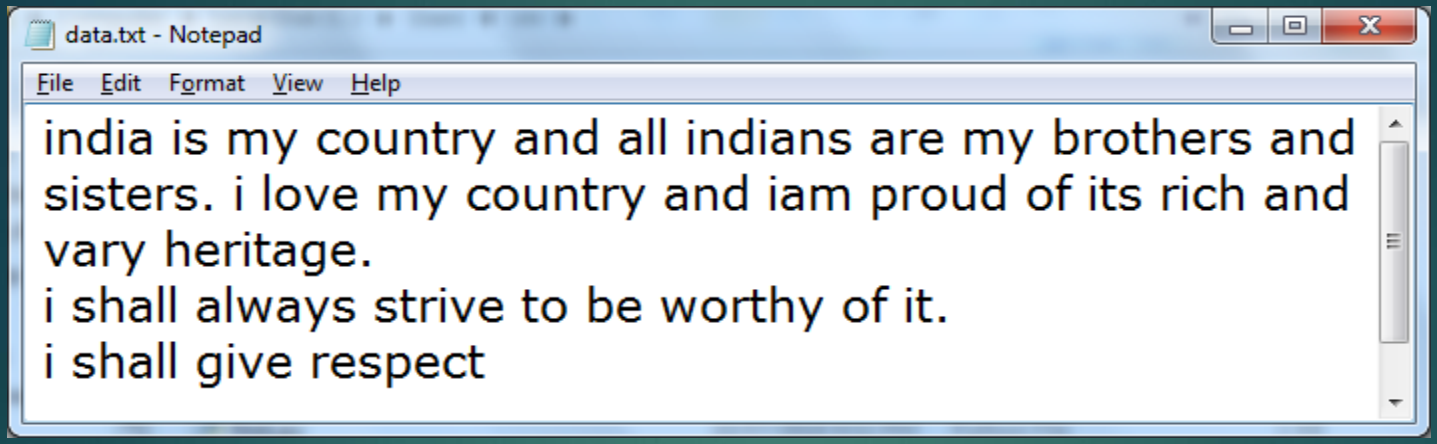

```
# program to read line
myfile = open("data.txt","r")
line1 = myfile.readline()
line2 = myfile.readline()
line3 = myfile.readline(10)
print(line1,end='')
print(line2,end='')
print(line3)
```

```
(base) C:\Users\vin>python thirdfile.py
india is my country and all indians are my brothers and sisters. i love my count
ry and iam proud of its rich and vary heritage.
i shall always strive to be worthy of it.
i shall gi
```

HAVE YOU NOTICED THE DIFFERENCE IN OUTPUT FROM PREVIOUS OUPUT?

Example-4: reading line by readline() [multi-line]

SAMPLE FILE

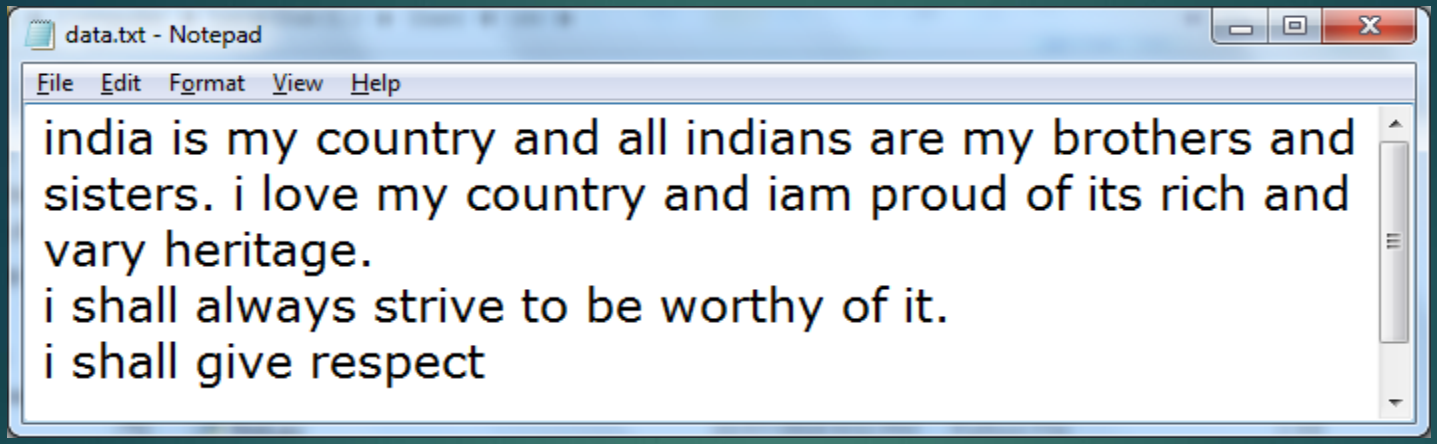



```
myfile = open("data.txt","r")
str=" "
while str:
    str = myfile.readline()
    print(str,end='')
myfile.close()
```

```
india is my country and all indians are my brothers and sisters. i love my count
ry and iam proud of its rich and vary heritage.
i shall always strive to be worthy of it.
i shall give respect
```

Example-5: reading line using for Loop

SAMPLE FILE

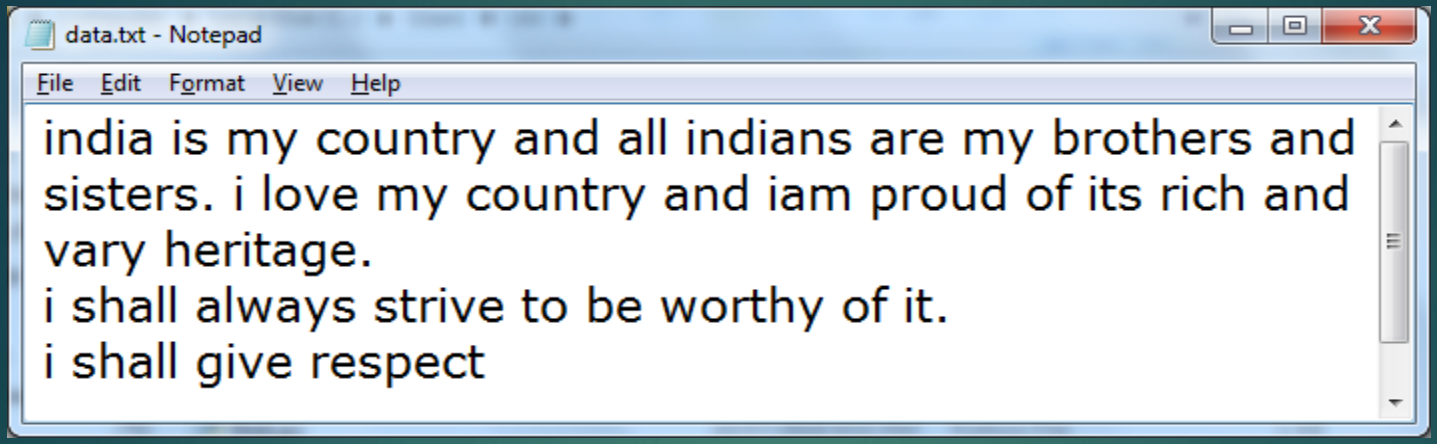



```
myfile = open("data.txt","r")
for str in myfile:
    print(str,end='')
```

```
india is my country and all indians are my brothers and sisters. i love my count
ry and iam proud of its rich and vary heritage.
i shall always strive to be worthy of it.
i shall give respect
```

Example-6: Calculating size of file with and without using blank lines

SAMPLE FILE

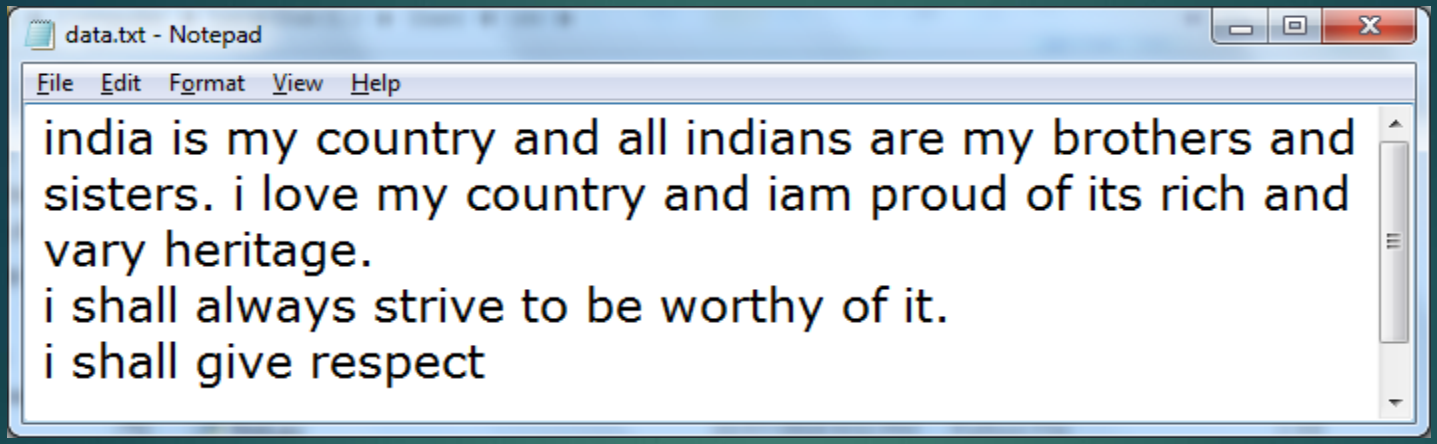



```
myfile = open("data.txt","r")
str1=" "
size=0
tsize=0
while str1:
    str1=myfile.readline()
    tsize = tsize + len(str1)
    size = size + len(str1.strip())
print("Total Size :",tsize)
print("Size after removing all EOL and blank lines :",size)
myfile.close()
```

```
Total Size : 191
Size after removing all EOL and blank lines : 188
```

Example7: readlines()

SAMPLE FILE



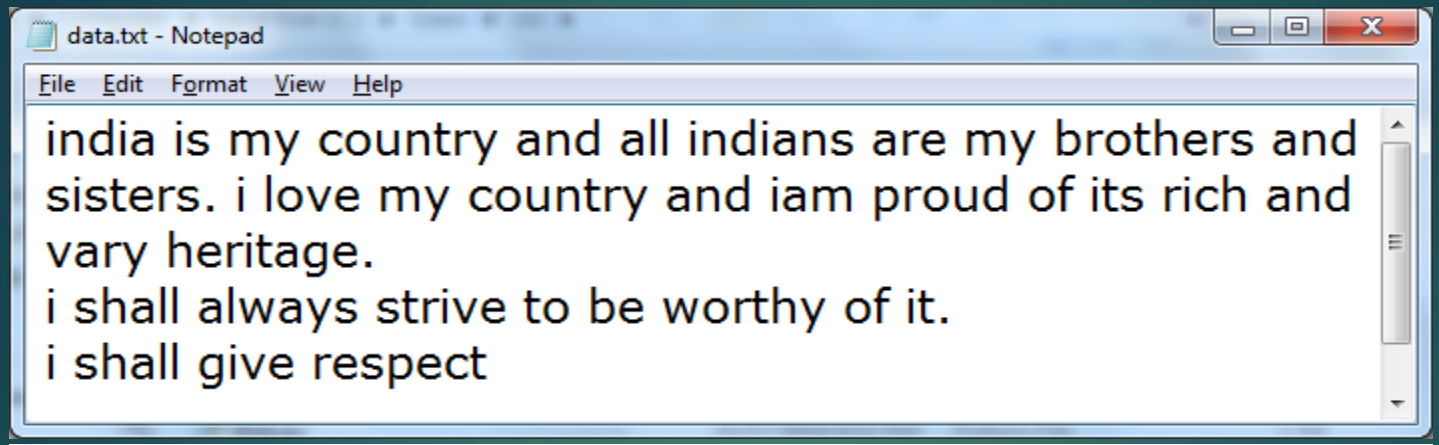

```
myfile = open ("data.txt","r")
contents = myfile.readlines()
print(contents)
print("First line is :",contents[0])
print("Last line is :",contents[len(contents)-1])
```

```
['india is my country and all indians are my brothers and sisters. i love my cou
ntry and iam proud of its rich and vary heritage.\n', 'i shall always strive to
be worthy of it.\n', 'i shall give respect\n']
First line is : india is my country and all indians are my brothers and sisters.
 i love my country and iam proud of its rich and vary heritage.

Last line is : i shall give respect
```

Example-8 : counting size of file in bytes and count number of lines

SAMPLE FILE




```
myfile = open("data.txt","r")
str = myfile.read()
size = len(str)
print("Size of file in bytes :",size)
myfile.close()
```

```
Size of file in bytes : 191
```

```
myfile = open("data.txt","r")
str = myfile.readlines()
lines = len(str)
print("Number of lines in file :",lines)
myfile.close()
```

```
Number of lines in file : 3
```

Questions…

1. WAP to read the content of file and count how many times letter 'a' comes in file
2. WAP to read the content of file and display 'I' in place of 'E' while displaying the content of file all other characters should appear as it is.
3. WAP to read the content of file and display how many upper case characters and digits are present.
4. WAP to read the content of file and count how many vowels in it.
5(a). Write a statement in Python to open text file NEWS.TXT so that new content can be written
5(b). Write a statement in Python to open text file NEWS.TXT so that existing content can be read
5(c). Write a statement in Python to open text file NEWS.TXT so that new content can be written after the previous content.
6. Write a function ISTOUPCOUNT() in python to read the content of file WRITER.TXT, to count and display the occurance of IS, TO, UP.

For example if the content of file is:

IT IS UP TO US TO TAKE CARE OF OUR SURROUNDING. IT IS NOT POSSIBLE ONLY FOR THE GOVERNMENT TO TAKE RESPONSIBILITY.

The method/function should display

Count of IS, UP and TO is 6

7. Write a function COUNTAE() in python to read lines from text file WRITER.TXT, and display those lines, which are starting either with A or starting with E. For example if the content of file is :

A CLEAN ENVIRONMENT IS NECESSARY FOR OUR GOOD HEALTH.
WE SHOULD CLEAN OUR ENVIRONMENT.
EDUCATIONAL INSTITUTE SHOULD TAKE THE LEAD

The function should display:

A CLEAN ENVIRONMENT IS NECESSARY FOR OUR GOOD HEALTH.
EDUCATIONAL INSTITUTE SHOULD TAKE THE LEAD

Writing onto files

- After read operation, let us take an example of how to write data in disk files. Python provides functions:
  - write ()
  - writelines()
- The above functions are called by the file handle to write desired content.

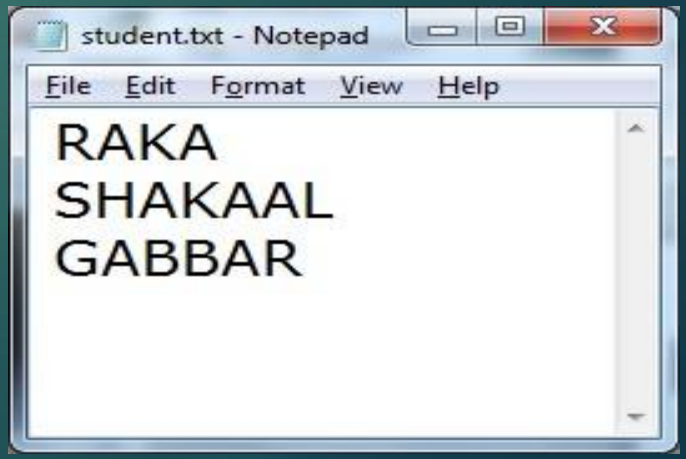
Name	Syntax	Description
write()	Filehandle.write(str1)	Writes string str1 to file referenced by filehandle
Writelines()	Filehandle.writelines(L)	Writes all string in List L as lines to file referenced byfilehandle.

Example-1: Write() using "w" mode

```
myfile = open("student.txt","w")
for i in range(3):
    name = input("Enter name to store :")
    myfile.write(name)
    myfile.write('\n')
myfile.close()
print("\nData Saved Successfully...")
```

```
Enter name to store :RAKA
Enter name to store :SHAKAAL
Enter name to store :GABBAR

Data Saved Successfully...
```



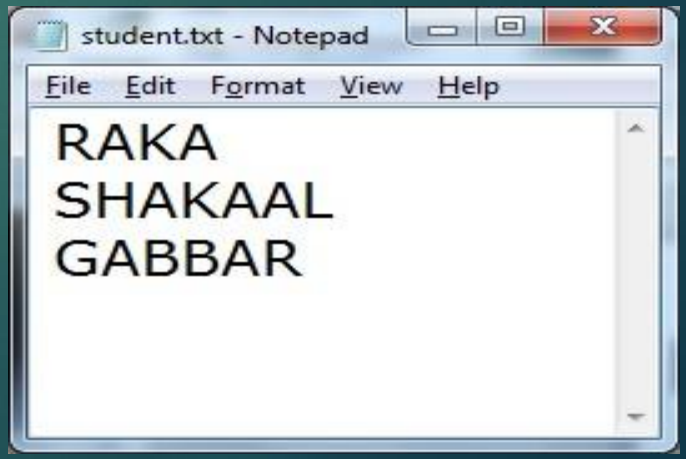
Example-1: Write using “w” mode

```
myfile = open("student.txt","w")
for i in range(3):
    name = input("Enter name to store :")
    myfile.write(name)
    myfile.write('\n')
myfile.close()
print("\nData Saved Successfully...")
```

```
Enter name to store :RAKA
Enter name to store :SHAKAAL
Enter name to store :GABBAR

Data Saved Successfully...
```

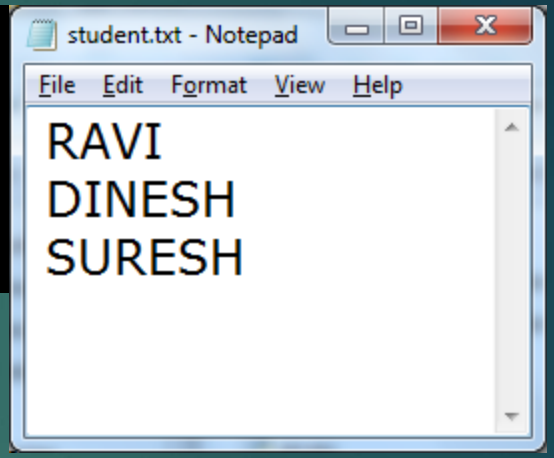
Lets run the same program again



Example-1: write() using "w" mode

```
Enter name to store :RAVI
Enter name to store :DINESH
Enter name to store :SURESH

Data Saved Successfully...
```



Now we can observe that while writing data to file using "w" mode the previous content of existing file will be overwritten and new content will be saved.

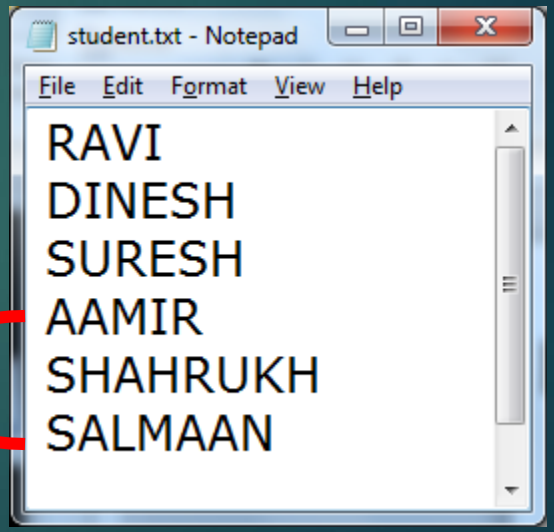
If we want to add new data without overwriting the previous content then we should write using "a" mode i.e. append mode.

Example-2: write() using "a" mode

```
myfile = open("student.txt","a")
for i in range(3):
    name = input("Enter name to store :")
    myfile.write(name)
    myfile.write('\n')
myfile.close()
print("\nData Saved Successfully...")
```

```
Enter name to store :AAMIR
Enter name to store :SHAHRUKH
Enter name to store :SALMAAN

Data Saved Successfully...
```

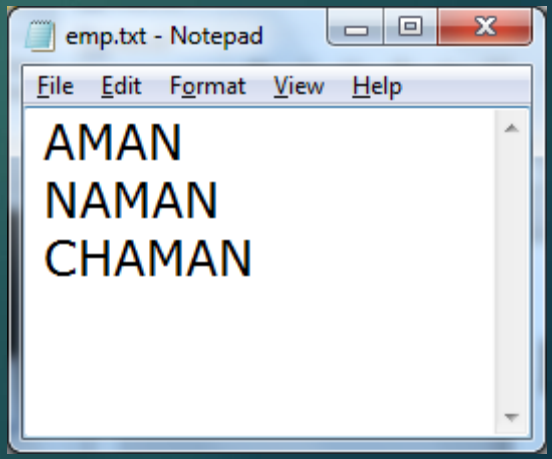


New content is added after previous content

Example-3: using writelines()

```
myfile = open("emp.txt","w")
mylist = []
for i in range(3):
    name=input("Enter employee name :")
    mylist.append(name+'\n')
myfile.writelines(mylist)
myfile.close()
print("Data Saved...")
```

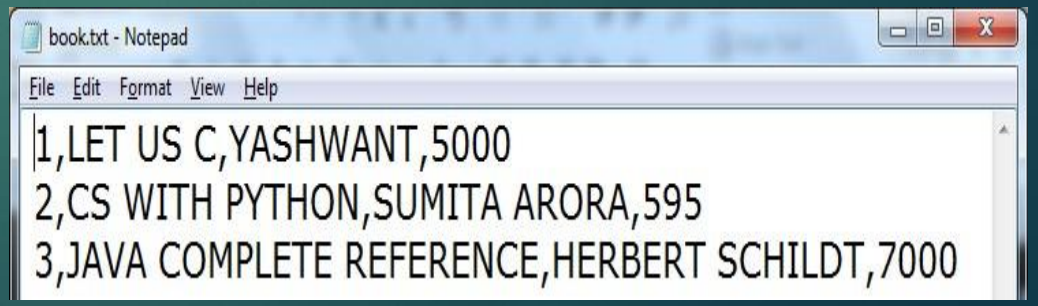
```
Enter employee name :AMAN
Enter employee name :NAMAN
Enter employee name :CHAMAN
Data Saved...
```



Example-4: Writing

```
myfile = open("book.txt","a")
ans='y'
while ans=='y':
    bno = int(input("Enter book number "))
    bname = input("Enter Book Name ")
    author = input("Enter Author Name  ")
    price = int(input("Enter Book Price "))
    brec = str(bno) + "," + bname +","+author+","+str(price)+"\n"
    myfile.write(brec)
    ans=input("Add More ?")
myfile.close()
```

```
Enter book number 1
Enter Book Name LET US C
Enter Author Name  YASHWANT
Enter Book Price 5000
Add More ?y
Enter book number 2
Enter Book Name CS WITH PYTHON
Enter Author Name  SUMITA ARORA
Enter Book Price 595
Add More ?y
Enter book number 3
Enter Book Name JAVA COMPLETE REFERENCE
Enter Author Name  HERBERT SCHILDT
Enter Book Price 7000
Add More ?n
```



Example: copy one file to another file

```
myfile1 = open("book.txt","r")
myfile2 = open("book_backup.txt","w")
str1=" "
while str1:
    str1 = myfile1.readline()
    myfile2.write(str1)
myfile1.close()
myfile2.close()
print("Copied Successfully...")
```

```
Copied Successfully...
```

book.txt - Notepad

File Edit Format View Help

1,LET US C,YASHWANT,5000
2,CS WITH PYTHON,SUMITA ARORA,595
3,JAVA COMPLETE REFERENCE,HERBERT SCHILDT,7000

book_backup.txt - Notepad

File Edit Format View Help

1,LET US C,YASHWANT,5000
2,CS WITH PYTHON,SUMITA ARORA,595
3,JAVA COMPLETE REFERENCE,HERBERT SCHILDT,7000

flush() function

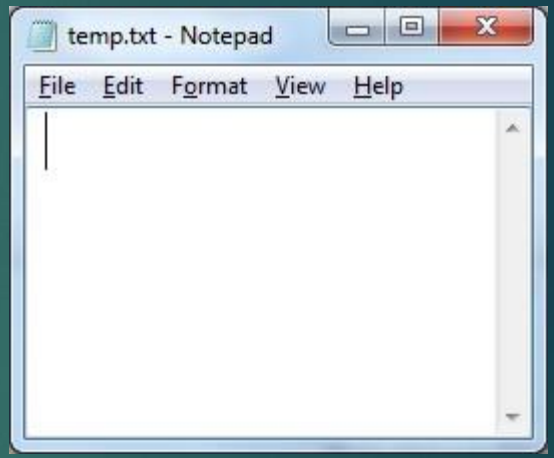
- When we write any data to file, python hold everything in buffer (temporary memory) and pushes it onto actual file later. If you want to force Python to write the content of buffer onto storage, you can use flush() function.
- Python automatically flushes the files when closing them i.e. it will be implicitly called by the close(), BUT if you want to flush before closing any file you can use flush()

Example: working of flush()

Without flush()

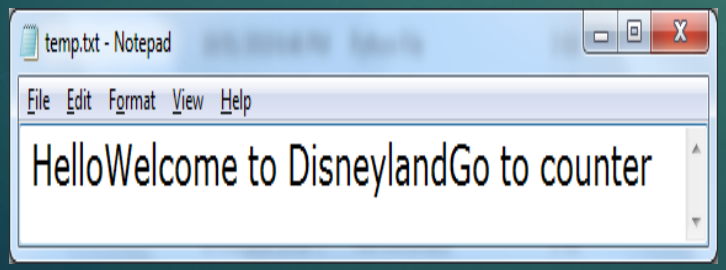
```
myfile = open("temp.txt","w+")
myfile.write("Hello")
myfile.write("Welcome to Disneyland")
n = input("press any key")
myfile.write("Go to counter")
myfile.close()
```

Nothing is in the file temp.txt



When you run the above code, program will stopped at “Press any key”, for time being don’t press any key and go to folder where file “temp.txt” is created an open it to see what is in the file till now

NOW PRESS ANY KEY….



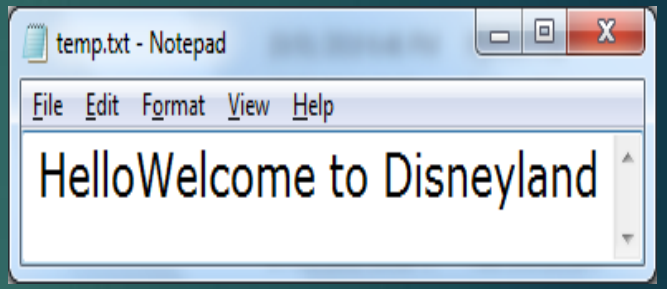
Now content is stored, because of close() function contents are flushed and pushed in file

Example: working of flush()

With flush()

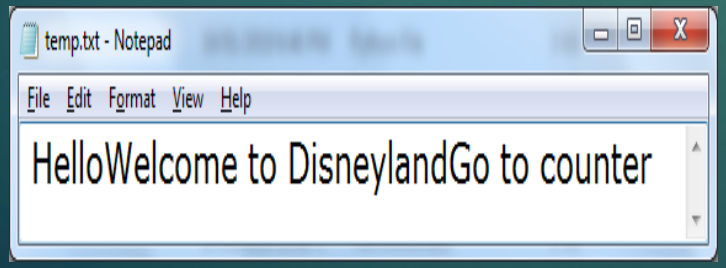
```
myfile = open("temp.txt","w+")
myfile.write("Hello")
myfile.write("Welcome to Disneyland")
myfile.flush()
n = input("press any key")
myfile.write("Go to counter")
myfile.close()
```

All contents before flush() are present in file



When you run the above code, program will stopped at “Press any key”, for time being don’t press any key and go to folder where file “temp.txt” is created an open it to see what is in the file till now

NOW PRESS ANY KEY….



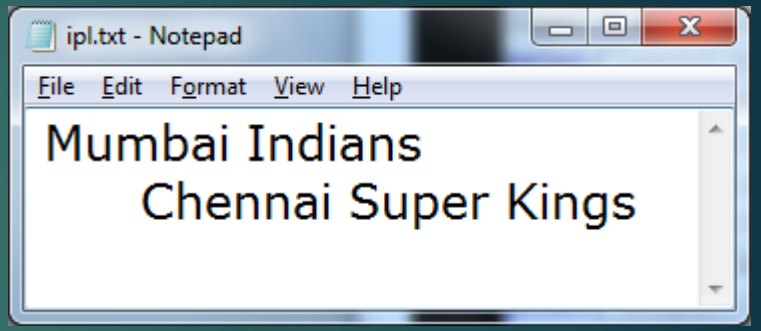
Rest of the content is written because of close(), contents are flushed and pushed in file.

Removing white space after reading all text.

- read() and readline() reads data from file and return it in the form of string and readlines() returns data in the form of list.
- All these read function also read leading and trailing whitespaces, new line characters. If you want to remove these characters you can use functions
  - strip() : removes the given character from both ends.
  - lstrip(): removes given character from left end
  - rstrip(): removes given character from right end

Example: strip(),lstrip(), rstrip()

```
myfile = open("ipl.txt")
line1 = myfile.readline()
print("Length of Line is :",len(line1))
line1 = line1.rstrip('\n')
print("Length of Line is :",len(line1))
line2 = myfile.readline()
print("Length of Line is :",len(line2))
line2 = line2.lstrip()
print("Length of Line is :",len(line2))
```



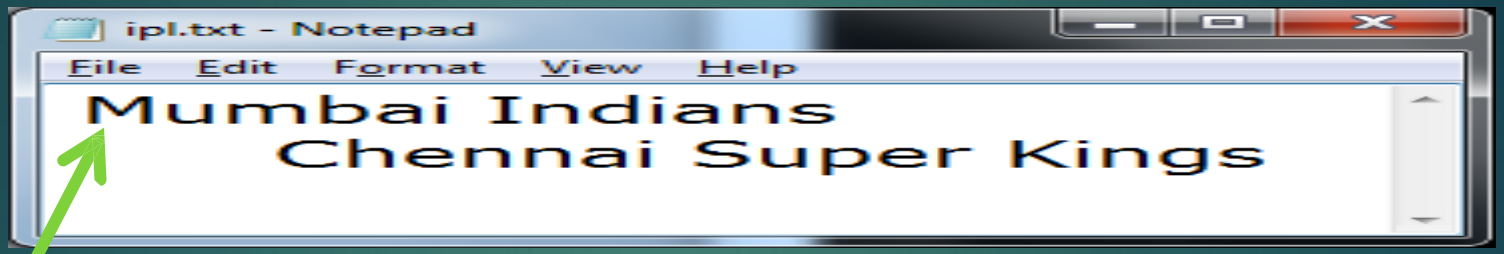

```
Length of Line is : 15
Length of Line is : 14
Length of Line is : 25
Length of Line is : 19
```

File Pointer

- Every file maintains a file pointer which tells the current position in the file where reading and writing operation willtake.
- When we perform any read/write operation two things happens:
  - The operation at the current position of file pointer
  - File pointer advances by the specified number of bytes.

Example

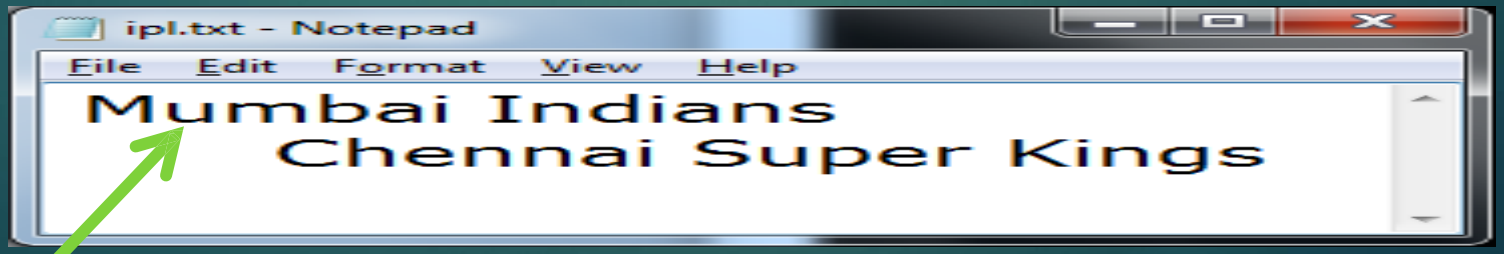
myfile = open("ipl.txt","r")



File pointer will be by default at first position i.e. first character

ch = myfile.read(1)

ch will store first character i.e. first character is consumed, and file pointer will move to next character



File Modes and Opening position of file pointer

FILE MODE	OPENING POSITION
r, r+, rb, rb+, r+b	Beginning of file
w, w+, wb, wb+, w+b	Beginning of file (overwrites the file if file already exists
a, ab, a+, ab+, a+b	At the end of file if file exists otherwise creates a new file

Standard INPUT, OUTPUT and ERROR STREAM

- Standard Input : Keyboard
- Standard Output : Monitor
- Standard error : Monitor

- Standard Input devices(stdin)reads from keyboard
- Standard output devices(stdout) display output on monitor
- Standard error devices(stderr) same as stdout but normally for errors only.

Standard INPUT, OUTPUT and ERROR STREAM.

- The standard devices are implemented as files called standard streams in Python and we can use them by using **sys** module.
- After importing **sys** module we can use standard streams **stdin, stdout, stderr**

```
import sys
myfile = open("ipl.txt","r")
line1 = myfile.readline()
line2 = myfile.readline()
sys.stdout.write(line1+'\n')
sys.stdout.write(line2+'\n')
sys.stderr.write("No error found\n")
myfile.close()
```

```
Mumbai Indians

        Chennai Super Kings
No error found
```

"with" statement

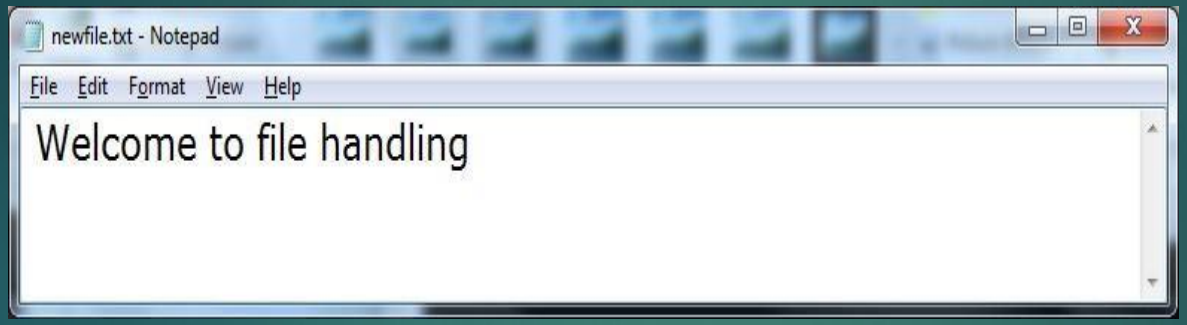
- Python's "with" statement for file handling is very handy when you have two related operations which you would like to execute as a pair, with a block of code in between:

```
with open(filename[, mode]) as filehandle:
    file_manipulation_statement
```

- The advantage of "with" is it will automatically close the file after nested block of code. It guarantees to close the file how nested block exits even if any run time error occurs

Example

```
with open("newfile.txt","w") as mf:
    mf.write("Welcome to file handling")
```



Binary file operations

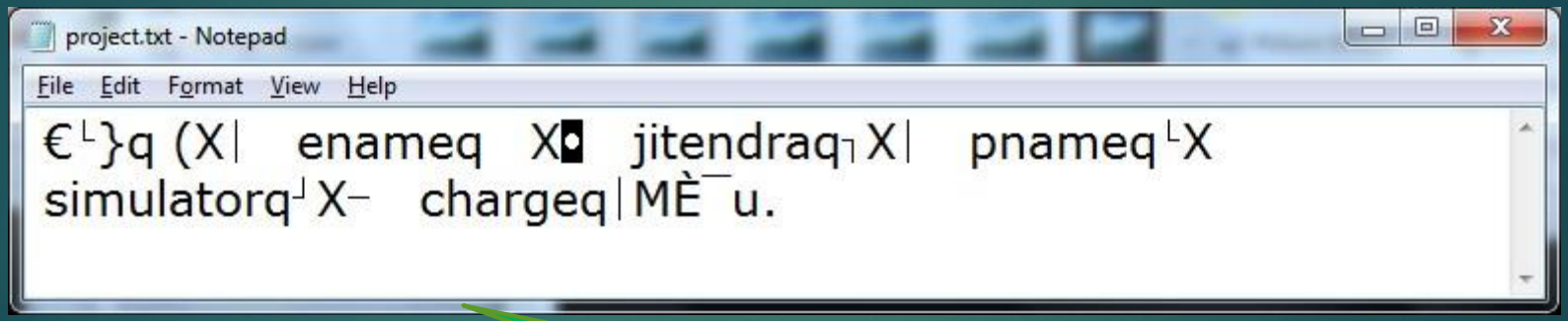
- If we want to write a structure such as list or dictionary to a file and read it subsequently we need to use the Python module **pickle**. **Pickling** is the process of converting structure to a byte stream before writing to a file and while reading the content of file a reverse process called **Unpickling** is used to convert the byte stream back to the original format.

Steps to perform binary file operations

- First we need to import the module called pickle.
- This module provides 2 main functions:
  - dump() : to write the object in file which is loaded in binary mode
    - Syntax : dump(object_to_write, filehandle)
  - load() : dumped data can be read from file using load() i.e. it is used to read object from pickle file.
    - Syntax: object =load(filehandle)

Example: dump()

```
# program to demonstrate binary file operations
import pickle
myfile = open("project.txt","wb")
dict1 = {'ename':'jitendra','pname':'simulator','charge':45000}
pickle.dump(dict1,myfile)
myfile.close()
```



See the content is some kind of encrypted format, and it is not in complete readable form

Example: load()

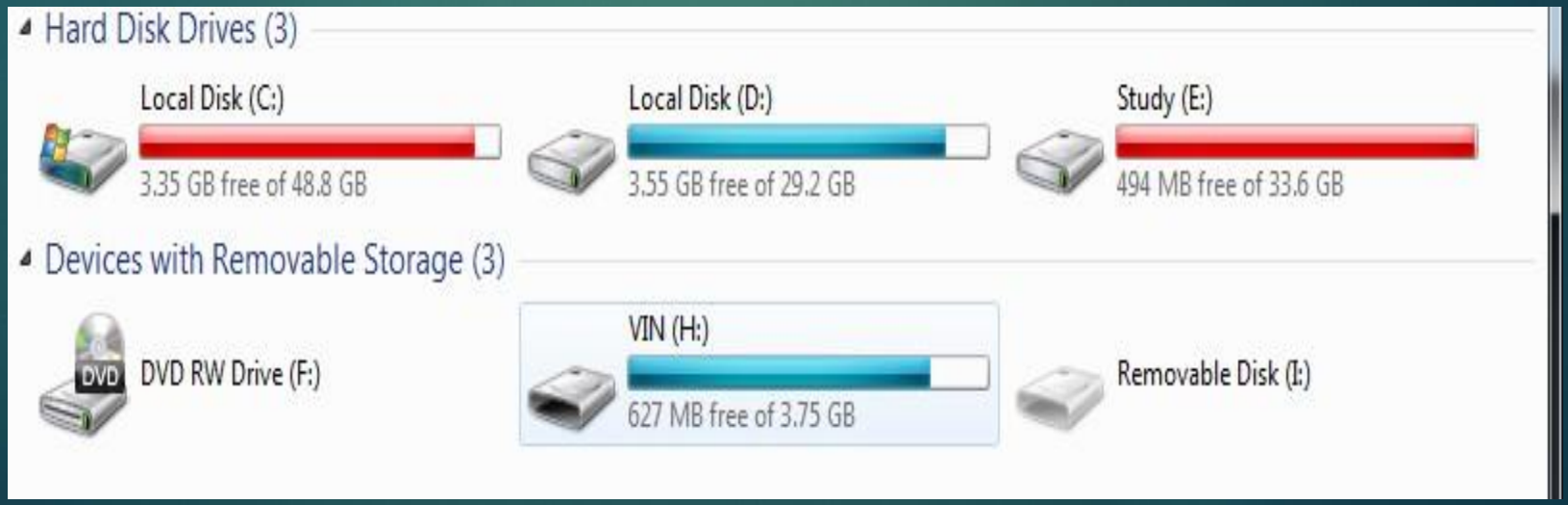
```
# program to demonstrate binary file operations
import pickle
mf = open("project.txt","wb")
#dict1 = pickle.load(mf)
#myfile.close()
dict1 = {1:"a",2:"b",3:"c"}
pickle.dump(dict1,mf)
mf.close()
```

```
# program to demonstrate binary file operations
import pickle
mf = open("project.txt","rb")
dict1 = pickle.load(mf)
print(dict1)
print("Item 1 is ",dict1[1])
mf.close()
```

```
{1: 'a', 2: 'b', 3: 'c'}
Item 1 is  a
```

Absolute Vs Relative PATH

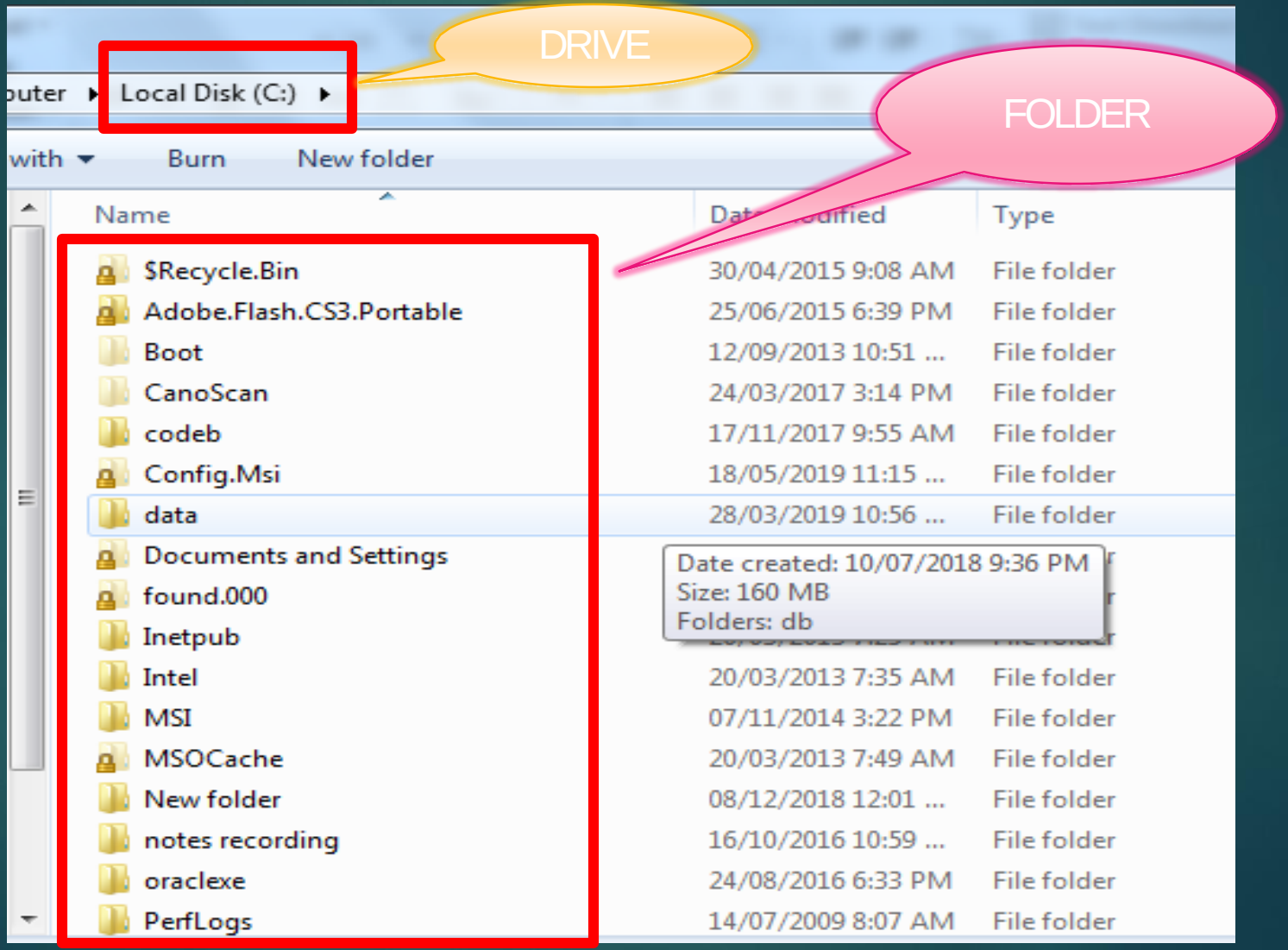
- To understand PATH we must be familiar with the terms: DRIVE, FOLDER/DIRECTORY, FILES.
- Our hard disk is logically divided into many parts called DRIVES like C DRIVE, D DRIVE etc.



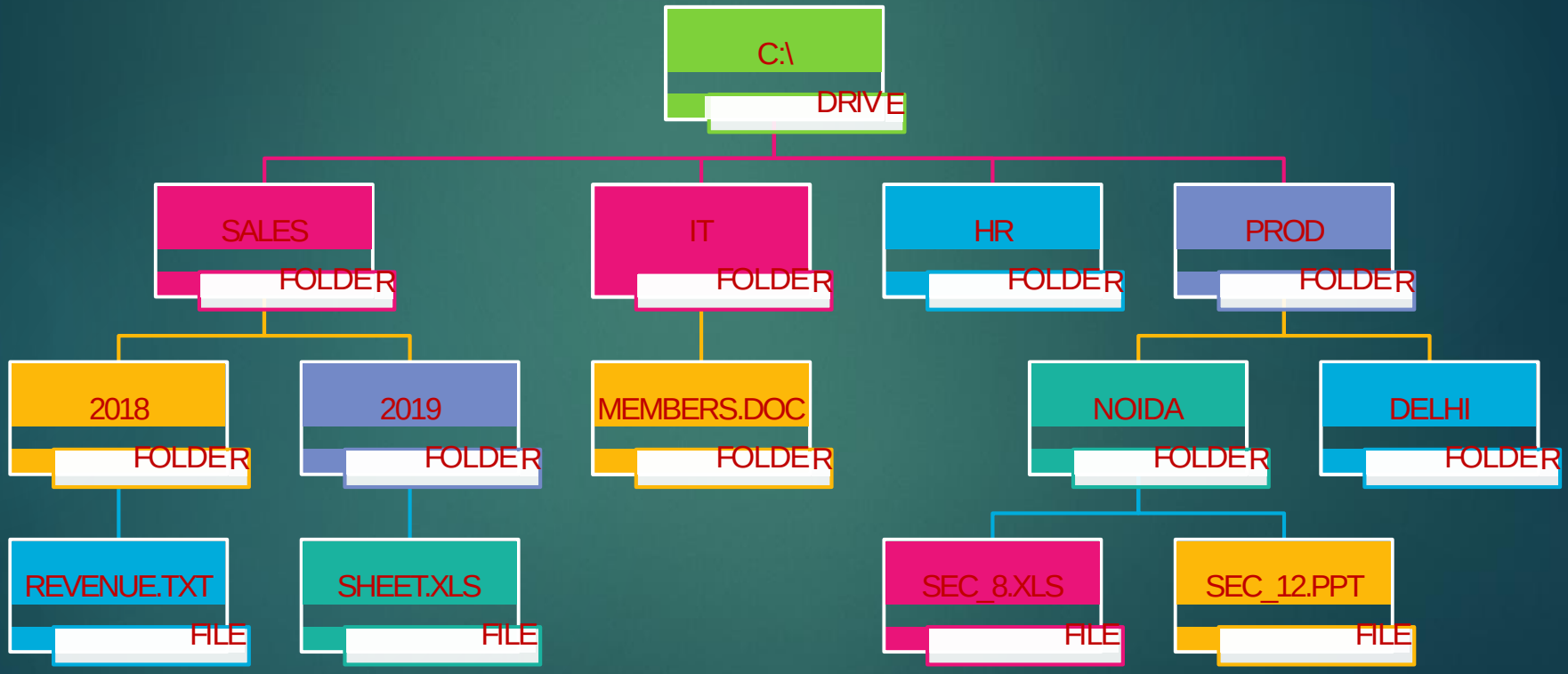
Absolute Vs Relative PATH

- The drive is the main container in which we put everything to store.
- The naming format is : DRIVE_LETTER:
- For e.g. C: , D:
- Drive is also known as **ROOT DIRECTORY.**
- Drive contains Folder and Files.
- Folder contains sub-folders or files
- Files are the actual data container.

Absolute Vs Relative PATH



DRIVE/FOLDER/FILE HIERARCHY



Absolute Path

- Absolute path is the full address of any file or folder from the Drive i.e. from ROOT FOLDER. It is like:

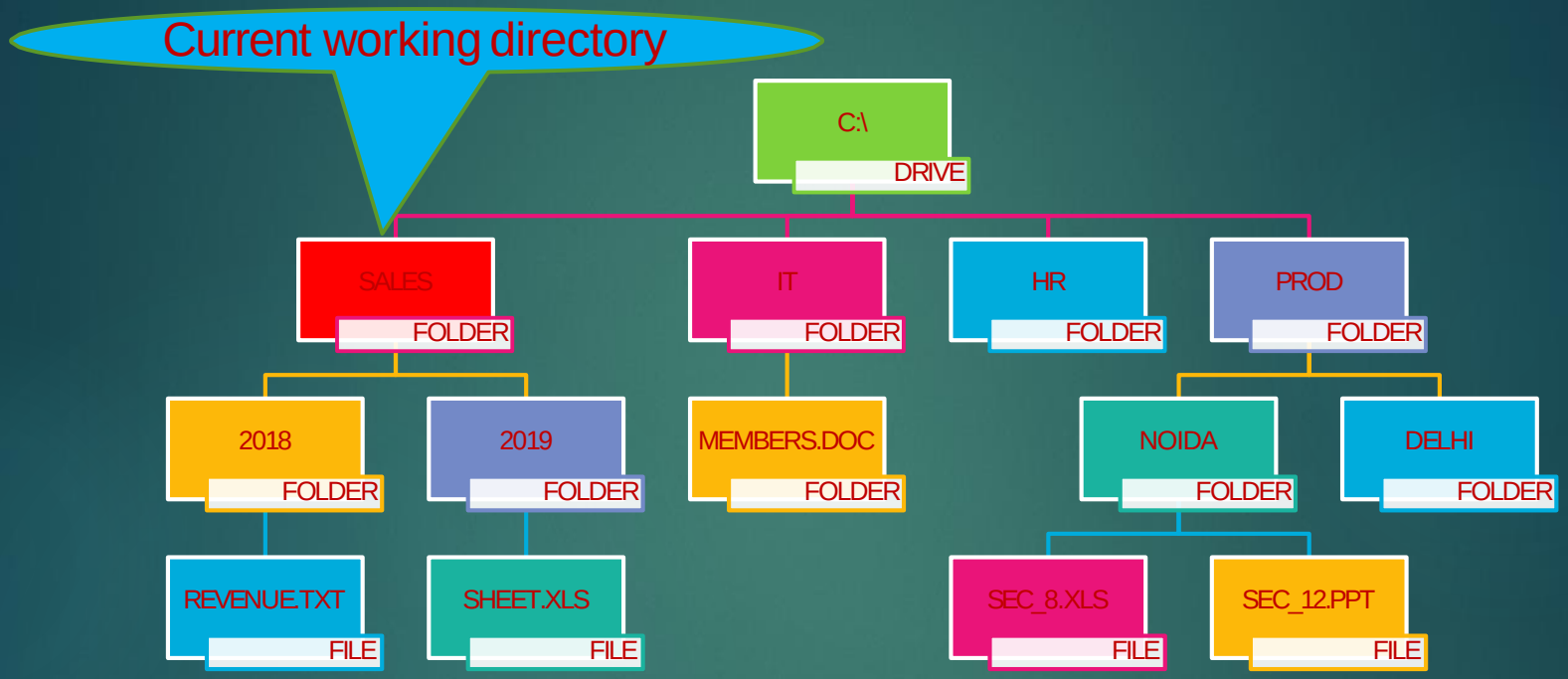
  Drive_Name:\Folder\Folder…\filename

- For e.g. the Absolute path of file REVENUE.TXT will be
  - **C:\SALES\2018\REVENUE.TXT**
- Absolute path of SEC_12.PPT is
  - **C:\PROD\NOIDA\Sec_12.ppt**

Relative Path

- Relative Path is the location of file/folder from the current folder. To use Relative path special symbols are:
  - Single Dot (.) : single dot (.) refers to current folder.
  - Double Dot (..) : double dot (..) refers to parent folder
  - Backslash (\ ) : first backslash before (.) and double dot(..) refers to ROOT folder.

Relative addressing

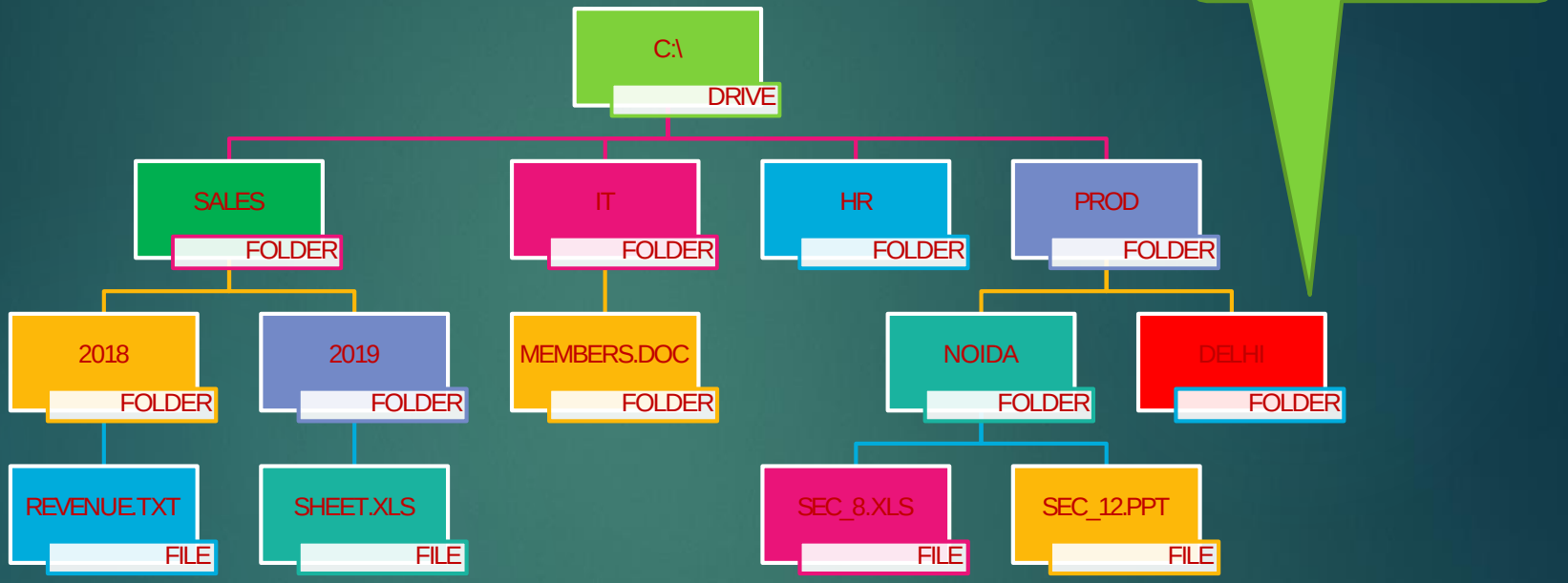


SUPPOSE CURRENT WORKING DIRECTORY IS : **SALES**

WE WANT TO ACCESS SHEET.XLS FILE, THEN RELATIVE ADDRESS WILL BE

.\2019\SHEET.XLS

Relative addressing



SUPPOSE CURRENT WORKING DIRECTORY IS : **DELHI**

WE WANT TO ACCESS **SEC_8.XLS** FILE, THEN RELATIVE ADDRESS WILL BE

..\NOIDA**SEC_8.XLS**

Getting name of current working directory

```
import os
pwd = os.getcwd()
print("Current Directory :",pwd)
```

Thank U