WORKING WITH PYTHON FILES:
File is collection of related information. In Python, a file operation takes place in the following order.
1 Open a file 2 Read or write a file
3 Append a file 4 Close the file

Opening a file
Python has a built-in function open() to open a file. This function returns a file object. It returns a "file handle" - a variable used to perform operations on the file. file handle has.!
1.Open2.Write3.Read4. Close

Syntax
FileObject=open(file_name [, access_mode][, buffering])

Parameter details:
file_name: It contains the name of the file.
access_mode: Default file access mode is read (r).
buffering: The buffering value 0, no buffering takes place. If 1, line buffering is performed.

Example:
```
FileObj=open("AnyName.txt")#relative path
FileObj=open("C:\\Python33\\AnyName.txt") #(AbsolutePath)
```

File Different Modes:
ModesDescription
rOpens a file for reading only. (default)
b Opens in binary mode.
r+Opens a file for both reading & writing.
rb+Opens a file for both reading & writing in binary format
wOpens a file for writing only.
aOpens a file for appending.
a+Opens a file for both appending and reading.
't' Opens in text mode. (default)
x Open a file for exclusive creation. If file already exists Operation fails.

The file Object Attributes
AttributeDescription
file.nameReturns name of the file
file.modeReturns access mode
writable() Returns boolean value
readable() Retruns boolean value
file.closedRetruns boolean value

The best way to do this is using the with statement. This ensures that the file is closed when the block inside with is existed. We don't need to explicitly call the close() method. It done Implicitly.

Syntax:
```
with open("MyFile.txt",mode='r',encoding = 'utf-8') as MyFObj:
   # Perform Required File Operations
```

Example:
```
try:
```

```
    with open("MyFile.txt",mode='r',encoding='utf-8') as MyFile:
        print(MyFile.name)
        print(MyFile.mode)
        print(MyFile.closed)
        print(MyFile.readable())
        print(MyFile.writable())
except IOError:
    print("SorryFileNotExisted")
finally:
    print("FinallyBlockSuccess")
```

Writing data into a File
In order to write into a file we need to open it in write 'w', append 'a' or exclusive creation 'x' mode.

NOTE:
We need to be careful with the 'w' mode as it will overwrite into the file if it already exists.

write() Method:
Using this method we can write a specific line in the file.

Syntax:
```
Object.write("Text")
```

writelines() Method:
Using this method we can write a specific multiple lines in the file.

Syntax:
```
Object.writelines("Text lines")
```

Example:
```
try:
    with open("MyFile.txt",mode='w',encoding='utf-8') as MyFileObj:
        MyFileObj.writable()
        MyFileObj.write("Welcome to PYTHON File Operations\n")
        MyFileObj.writelines("""Multiple lines are
                             Writing inside the A file.
                             Thank U""")
except IOError:
    print("SorryFileUnableToCreate")
    print("DiskWriteProtected")
finally:
    print("FinallyBlockExecutedSuccessfully")
    print("FileOperationsSuccess")
```

Example: Absolutepath or FullPath or LocalPath:
```
with
open("C:\\Users\\admin\\Desktop\\Data\\MyFile.txt",mode='w',encoding="
utf-8") as MyFile:
    MyFile.writelines("हिन्दी हिन्दी हिन्दी हिन्दी\n")
    MyFile.writelines("తెలుగు తెలుగు తెలుగు తెలుగు\n")
    MyFile.writelines("Thank U")
print("FileCreatedSuccessfully")
```

Reading Data From a File
To read the content of a file, we must open the read mode. There are various methods to read a file.

1. read() method:
It can read the data depends on size of characters read(size). If size parameter is not specified, it reads and returns complete the file.

Syntax
fileObject.read([count]);

readline(): to read individual lines of a file
Syntax:
fileObject.readline()

readlines()==> To read all lines into a list
Syntax:
fileObject.readlines()

Example:
```
with open("python.txt",'r',encoding = 'utf-8')as fi:
    print(fi.read(3))    # read the first 3 chars. data
    print(fi.read())     # read in the rest till end of file
    print(fi.read())  # further reading returns empty sting
fi.close()
```

Example:
```
with open("python.txt",'r',encoding = 'utf-8') as fi:
    print(fi.read(5))
    str = fi.read(10);
    print("Read Strting is: " ,str)
```

Example: Reading the Whole File
```
with open("python.txt",'r',encoding = 'utf-8') as fi:
    print(fi.read())
fi.close()
```

Example:
```
#Reading the file and displaying number of characters
with open("Hai.txt",'r',encoding='utf-8') as fi:
    chars=fi.read()
print("Number of characters are: ", len(chars))
```

Example:
```
with open("python.txt",'r',encoding = 'utf-8')as fi:
    print(fi.readline())
```

Example:
```
with open("python.txt",'r',encoding = 'utf-8')as fi:
    print(fi.readlines())
```

Example:
```
#Reading the file and displaying number of Lines
with open("YourFile.txt",'r',encoding='utf-8') as fi:
    lines=fi.readlines()
print("Number of characters are: ", len(lines))
```

NOTE: All these reading method return empty values when end of file (EOF) is reached.

Example:

```
with open("MyFile.txt",mode='r',encoding='utf-8') as MyFile:
    lines=MyFile.readlines()
    for line in lines:
        print(line,end='')
MyFile.close()
```

Example:

```
#Counting Number of lines in a file
with open("YourFile.txt",'r',encoding="utf-8") as lcount:
    count = 0
    for line in lcount:
        count = count + 1
print('Number of Lines in a File:', count)
```

Example:

```
Number of Spaces in a file:
myfile=open("MyFile.txt")
c=myfile.readlines()
intialline=len(c)
finalline=0
for i in c:
    z=i.split(" ")
    for j in z:
        finalline+=1
totalnumofspace=finalline-intialline
print(totalnumofspace)
```

Example:

```
fhand = open('Hello.txt','r')
for line in fhand:
   if line.startswith('From:') :
        print(line)
```

Example:

```
import os,sys
fname=input("Enter Any File Name: ")
if os.path.isfile(fname):
    print("File Existed:",fname)
    f=open(fname,mode="r")
else:
    print("Sorry File does not exist:",fname)
    sys.exit()
lcount=wcount=ccount=0
for line in f:
    lcount=lcount+1
    ccount=ccount+len(line)
    words=line.split()
    wcount=wcount+len(words)
print("Number of Lines:",lcount)
print("Number of Words:",wcount)
print("Number of Characters:",ccount)
```

Appending a file:

We can able to append a file with the help of 'a'. Append means add the data to an existing file, Under existing

data.

Example:

```
with open("Hai.txt",'a',encoding="utf-8") as file:
    file.write("Appending a File\n")
    file.write("Thank U\n")
print("Successfully File Appended")
```

File cursor Positions

The tell() method tells you the current cursor position within the file.

Syntax:

```
fileObject.tell()
```

Example:

```
with open("python.txt",'r',encoding = 'utf-8')as fi:
    print(fi.read(5))
    str = fi.read(10);
    print("Read Strting is: " ,str)
    position=fi.tell()
print("The Position of the cursor is: ",position)
```

seek() Method:

This method changes the cursor position in the file.

Syntax:

```
seek(offset[, from])
```

1 offset represents the number of positions
2 from
0---->From beginning of file(default value)
1---->From current position
2--->From end of the file

Note: Python 2 supports all 3 values but Python 3 supports only zero.

Example:

```
with open("python.txt",'r',encoding = 'utf-8')as fi:
    print(fi.read(5))
    str = fi.read(10);
    print("Read Strting is: " ,str)
    position=fi.seek(0,0)
print("The Position of the cursor is: ",position)
```

Example:

```
with open("python.txt",'r',encoding = 'utf-8')as fi:
    print(fi.read(5))
    str = fi.read(10);
    print("Read Strting is: " ,str)
    position = fi.seek(0, 0);
    str = fi.read(10);
print("Again read String is : ", str)
fi.close()
```

Example with for loop:

```
with open("python.txt",'r',encoding = 'utf-8')as fi:
    for line in fi:
        print(line, end = '')
```

Handling Binary Data:
It is very common requirement to read or write binary data like images,video files,audio files etc.

Example:

```
MyFile1=open("django.png",mode="rb")
MyFile2=open("dj.png",mode="wb")
Byts=MyFile1.read()
MyFile2.write(Byts)
print("New Image is : dj.png")
```

Working with Zipping & Unzipping Files:
In PYTHON we can do zip and unzip files. Zip common featers are:
1.To improve memory utilization
2.We can reduce transfor time in network
3.We can improve performance of transfor files

To create Zip file:
We have to create ZipFile class object with name of the zip file,mode and constant ZIP_DEFLATED. This constant represents we are creating zip file.

Syntax:

```
Obj=ZipFile("ZipFileName",mode,"ZipType")
```

Example2:

```
from zipfile import *
ZFile=ZipFile("MyFile.zip","w",ZIP_DEFLATED)
ZFile.write("MyFile.txt")
ZFile.write("MyFile.csv")
ZFile.write("JSON.txt")
ZFile.close()
```

Unzip Operations:
ZIP_STORED represents unzip operation. This is default value and hence we are not required to specify. Once we created ZipFile object for unzip operation,we can get all file names present in that zip file by using namelist() method.

Syntax:

```
FileObj = ZipFile("MyFiles.zip","r",ZIP_STORED)
names = FileObj.namelist()
```

Example:

```
from zipfile import *
ZFile=ZipFile("MyFile.zip","r",ZIP_STORED)
names=ZFile.namelist()
for name in names:
    print("File Name: ",name)
```

Working PYTHON OS Module:

The rename() Method
It takes two arguments, the current filename and the new filename.

Syntax
os.rename(current_file_name, new_file_name)

Example:
```
import os
os.rename("Anaconda.txt","python.txt")
```

The remove() Method
It is used to delete files.

Syntax
os.remove(file_name)

Example
```
import os
os.remove("python.txt")
```

The mkdir() Method
It is used to create directories in the current location.

Syntax
os.mkdir("newdir")

Example
```
import os
os.mkdir("PYTHON")
```

The chdir() Method
It is used to change the current directory.

Syntax
os.chdir("Existingdir")

walk():
To display all contents of Current working directory including sub directories

Example:
```
import os
for dirpath,dirnames,filenames in os.walk('.'):
    print("Current Directory Path:",dirpath)
    print("Directories:",dirnames)
    print("Files:",filenames)
    print()
```

Example:Display all statistics of file:
```
import os
AllStat=os.stat("MyFile.txt")
print(AllStat)
```

Shutdown and Restart Computer in Python
To shutdown and restart your computer or pc or laptop using python code, you have to first import os library and then use os.system() function with the code "shutdown /s /t 1" and "shutdown /r /t 1" to

shutdown and restart your computer in a second.

NOTE:
Caution - Make sure to close all the program before running any of the below program as these program immediately shutdown or restart your computer.

Example:
```
import os
check = input("Shutdown your computer? (Y/N): ")
if check == 'N':
    exit()
else:
    os.system("shutdown /s /t 1")
```

Example:
```
import os;
check = input("Shutdown your computer? (Y/N): ")
if check == 'n':
    exit();
else:
    os.system("shutdown /r /t 1");
```

Example:
```
# Python Program - Shutdown and Restart Computer

import os;
print("1. Shutdown Computer");
print("2. Restart Computer");
print("3. Exit");
choice = int(input("\nEnter your choice: "));
if(choice>=1 and choice<=2):
    if choice == 1:
        os.system("shutdown /s /t 1");
    else:
        os.system("shutdown /r /t 1");
else:
    exit();
```

print():
It prints the given object to the standard output device (screen) or to the text stream file.

Syntax
```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

1 objects/values - object to the printed. * indicates that there may be more than one object or value
2 sep - objects are separated by sep. Default value: ' '
3 end - end is printed at last
4. file - must be an object with write(string) method. If omitted it, sys.stdout will be used which prints objects on the screen.
5. flush - If True, the stream is forcibly flushed. Default value: False

Example:
```
Sfile=open("MyFile.txt",mode='w')
```

```
print("Subba Raju Sir: ",file=Sfile)
print("Data Scientist: ",file=Sfile)
Sfile.close()
```

Example:

```
from time import sleep
#Output is flushed here
print("Hello, world!", end='', flush= True)
sleep(5)
print("Bye!!!")
#Output is not flushed here
print("HelpMe", end='')
sleep(5)
print("Okay!!!")
```

NOTE:
See the output - "Hello, world" and "Bye!!!" are printing correctly because before sleep(5) the print() is flushing but "HelpMe" and "Okay!!!" are printing together, because print() is not flushing.