File Handling :

The key function for working with files in Python is the open() function.
The open() function takes two parameters; filename, and mode.
There are four different methods (modes) for opening a file:

Variable name = open (“file name ” , “options”)

"r" - Read - Default value. Opens a file for reading, error if the file does not exist
"a" - Append - Opens a file for appending, creates the file if it does not exist
"w" - Write - Opens a file for writing, creates the file if it does not exist
"x" - Create - Creates the specified file, returns an error if the file exists

Read files :

Variable name = open(“filename”,”r”)

shuhari@debian: ~

```
>>> demo = open ('demo.txt','r')
>>> print(demo.read())
Hi demo file1
second line

>>>
```

To read by character :

shuhari@debian: ~

```
>>> demo = open ('demo.txt','r')
>>> print(demo.read())
Hi demo file1
second line

>>> demo.close()
>>> demo = open ('demo.txt','r')
>>> print(demo.read(2))
Hi
>>> print(demo.read(1))

>>> print(demo.read(3))
dem
>>> print(demo.read(4))
```

To read by line :

```
>>> demo.close()
>>> demo = open ('demo.txt','r')
>>> print(demo.readline())
Hi demo file1

>>> print(demo.readline(2))
se
>>> print(demo.readline())
cond line
```

By looping through the lines of the file, you can read the whole file, line by line:

```
>>> demo = open ('demo.txt','r')
>>> for elem in demo
  File "<stdin>", line 1
    for elem in demo
                   ^
SyntaxError: invalid syntax
>>> for elem in demo:
...     print(elem)
...
Hi demo file1

second line
```

To avoid space in between :

```
>>> demo.close()
>>> demo = open('demo.txt','r'
>>> for elem in demo:
...     print(elem,end='')
...
Hi demo file1
second line
```

Write to an existing content :

To write to an existing file, you must add a parameter to the open() function:
"a" - Append - will append to the end of the file
"w" - Write - will overwrite any existing content

Variable name = open("filename", 'a')

Append : it will append the content at end of the file :

shuhari@debian: ~

```
>>> demo = open('demo.txt','a')
>>> demo.write('appending the content')
21
>>> demo.close()
>>> demo = open('demo.txt','r')
>>> print(demo.read())
Hi demo file1
second line
appending the contentappending the content
>>>
```

After appending we need to close the file and then again pass the opetion ‘r’ and print it .

To overwrite :

For loop and print doesn’t work here

shuhari@debian: ~

```
>>> demo = open('demo.txt','w')
>>> demo.write('all content shall erased and its been overwritten')
49
>>> demo.close()
>>> demo = open('demo.txt','r')
>>> print(demo.read())
all content shall erased and its been overwritten
>>>
```

Create a New File

To create a new file in Python, use the open() method, with one of the following parameters:

Variable name = open (‘filename’, ‘options’)
"x" - Create - will create a file, returns an error if the file exist
"a" - Append - will create a file if the specified file does not exist
"w" - Write - will create a file if the specified file does not exist

shuhari@debian: ~

```
>>> demo = open ('demo.txt','x')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileExistsError: [Errno 17] File exists: 'demo.txt'
>>> demo = open ('demo.txt','a')
>>> demo = open ('demo.txt','w')
>>>
```

```
>>> demo = open ('newfile.txt','w')
>>> demo.write("hey buddy hi hwo are you")
24
>>> demo = open ('newfile.txt','r')
>>> print(demo.read())
hey buddy hi hwo are you
>>>
```

shuhari@debian: ~

```
>>> demo = open ('demo.txt','x')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileExistsError: [Errno 17] File exists: 'demo.txt'
>>> demo = open ('demo.txt','a')
>>> demo = open ('demo.txt','w')
>>> demo.close()
>>> demo = open ('newfile.txt','a')
>>> demo = open ('newfile.txt','r')
>>> demo = open ('newfile.txt','w')
>>> demo.write("hey buddy")
9
>>> print(demo.read())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
io.UnsupportedOperation: not readable
>>> demo.close()
>>> demo = open ('newfile.txt','r')
>>> print(demo.read())
hey buddy
>>>
```

Readline:

It reads the current line .

shuhari@debian: ~

```
  GNU nano 3.2                                    p2.py

#!/usr/bin/python3
import sys #sys module imported
def print_file(filename):     #defining a function
        f=open(filename)      #opening a file
        while True:             #while loop i.e while file exist folow to written block
                line=f.readline()       #passinf f.readline into a variable
                if not line:
                        break
                print(line,end="")
        f.close()
def main():
        print_file(sys.argv[1])

if __name__ == '__main__':
        main()
```

Output :

shuhari@debian: ~

```
shuhari@debian:~$ sudo nano p2.py
shuhari@debian:~$ sudo ./p2.py one.txt
A
B
C
D
E
F
shuhari@debian:~$
```

Each line is an element inside the list .

Readlines : it will print whole file and it takes whole conatin put it in list and then print hence every line is in list (readline convert into list)

```
shuhari@debian: ~
  GNU nano 3.2                                                    p3.py

#!/usr/bin/python3
import sys #sys module imported
def print_file(filename):        #defining a function
        f=open(filename)         #opening a file
        var1 = f.readlines()
        print(var1)
        print('*' * 20)
        for elem in var1:
                print(elem, end='')
        f.close()
def main():
        print_file(sys.argv[1])

if __name__ == '__main__':
        main()
```

Output :

```
shuhari@debian: ~
shuhari@debian:~$ sudo nano p3.py
shuhari@debian:~$ sudo ./p3.py one.txt
['A\n', 'B\n', 'C\n', 'D\n', 'E\n', 'F\n']
********************
A
B
C
D
E
```

Read :

It reads the file content as list

```
shuhari@debian: ~
  GNU nano 3.2                                                    p4.p

#!/usr/bin/python3
import sys #sys module imported
def print_file(filename):        #defining a function
        f=open(filename)         #opening a file
        var1 = f.read()
        print(var1,end="")
        f.close()
        print('-'*20)
        f1=open(filename,'r')
        print(f1.read())
def main():
        print_file(sys.argv[1])

if __name__ == '__main__':
        main()
```

Python dir handling :

Current working dir :

```
$sudo mkdir pylabs
$nano p1.py
#!/usr/bin/python3
Import os
def main():
        Print("pwd: ", os.getcwd())
If __name__=='__main__':
        main()
```

shuhari@debian: ~/pylabs

```
shuhari@debian:~/pylabs$ cat 1.py
#!/usr/bin/python3

import os
def main():
        print("Current working dir is : ", os.getcwd())

if __name__=='__main__':
        main()
shuhari@debian:~/pylabs$ sudo ./1.py
Current working dir is :  /home/shuhari/pylabs
shuhari@debian:~/pylabs$
```

List the contents inside dir :

```
#!/usr/bin/python3

import os,sys
def main():
      dir = sys.argv[1]
      content = os.listdir(dir)
      print("Content of dir are : ", content)

if __name__=='__main__':
      main()
```

Os.listdir returns list

```
shuhari@debian: ~/pylabs
shuhari@debian:~/pylabs$ sudo cat 2.py
#!/usr/bin/python3

import os,sys
def main():
        dir = sys.argv[1]
        content = os.listdir(dir)
        print("Content of dir are : ", content)

if __name__=='__main__':
        main()
shuhari@debian:~/pylabs$ sudo ./2.py /home/shuhari
Content of dir are :  ['two.txt', '.profile', 'p3.py', 'one.txt', 'wto.txt', 'three.txt', 'demo.txt', '
.local', 'p4.py', 'twoo.txt', 'pylabs', 'four.txt', '.python_history', '.bashrc', 'newfile.txt', '.bash
_logout', '.ssh', 'p5.py', 'p2.py', '.bash_history']
shuhari@debian:~/pylabs$ sudo ./2.py /home/
Content of dir are :  ['shuhari']
shuhari@debian:~/pylabs$ sudo ./2.py /home/shuhari/pylabs
Content of dir are :  ['1.py', '2.py']
shuhari@debian:~/pylabs$
```

For loop :

```
#!/usr/bin/python3

import os,sys
def main():
    dir = sys.argv[1]
    content = os.listdir(dir)
    for elem in content:
        print(elem)
if __name__=='__main__':
    main()
```

```
shuhari@debian: ~/pylabs
import os,sys
def main():
        dir = sys.argv[1]
        content = os.listdir(dir)
        for elem in content:
                print(elem)
if __name__=='__main__':
        main()
shuhari@debian:~/pylabs$ sudo ./3.py /home/shuhari/
two.txt
.profile
p3.py
one.txt
wto.txt
three.txt
demo.txt
.local
p4.py
twoo.txt
pylabs
four.txt
.python_history
.bashrc
newfile.txt
.bash_logout
.ssh
p5.py
p2.py
.bash_history
shuhari@debian:~/pylabs$ sudo ./3.py /home/shuhari/
```

Realative path :

shuhari@debian: ~/pylabs

```
  GNU nano 3.2                                          4.py

#!/usr/bin/python3

import os,sys
def main():
        dir = sys.argv[1]
        content = os.listdir(dir)
        for elem in content:
                relative_path = os.path.join(dir,elem)
                print("The relative path are : ",relative_path)
if __name__=='__main__':
        main()
```

Output :

If input is absoulte/relative output is also absolute/relative so this can be drawbackk or boon

shuhari@debian: ~/pylabs

```
shuhari@debian:~/pylabs$ sudo nano 4.py
shuhari@debian:~/pylabs$ sudo ./4.py /home/shuhari/pylabs
The relative path are :  /home/shuhari/pylabs/1.py
The relative path are :  /home/shuhari/pylabs/2.py
The relative path are :  /home/shuhari/pylabs/4.py
The relative path are :  /home/shuhari/pylabs/3.py
shuhari@debian:~/pylabs$
```

shuhari@debian: ~/pylabs

```
shuhari@debian:~/pylabs$ sudo ./4.py demo
The relative path are :  demo/file2.txt
The relative path are :  demo/file4.txt
The relative path are :  demo/file1.txt
The relative path are :  demo/file3.txt
shuhari@debian:~/pylabs$
```

Always a absolute path :

shuhari@debian: ~/pylabs

```
  GNU nano 3.2                                          5.py

#!/usr/bin/python3

import os,sys
def main():
        dir = sys.argv[1]
        content = os.listdir(dir)
        for elem in content:
                f_with_path = os.path.join(dir,elem)
                always_absolute_path = os.path.abspath(f_with_path)
                print("The absolute path are : ",always_absolute_path)
if __name__=='__main__':
        main()
```

shuhari@debian: ~/pylabs

```
shuhari@debian:~/pylabs$ sudo nano 5.py
shuhari@debian:~/pylabs$ sudo ./5.py demo
The absolute path are :  /home/shuhari/pylabs/demo/file2.txt
The absolute path are :  /home/shuhari/pylabs/demo/file4.txt
The absolute path are :  /home/shuhari/pylabs/demo/file1.txt
The absolute path are :  /home/shuhari/pylabs/demo/file3.txt
shuhari@debian:~/pylabs$
```

shuhari@debian: ~/pylabs

```
  GNU nano 3.2                                          5.py

#!/usr/bin/python3

import os,sys
def main():
        dir = sys.argv[1]
        content = os.listdir(dir)
        for elem in content:
                #f_with_path = os.path.join(dir,elem)
                always_absolute_path = os.path.abspath(dir)
                print("The absolute path are : ",always_absolute_path)
if __name__=='__main__':
        main()
```

shuhari@debian: ~/pylabs

```
shuhari@debian:~/pylabs$ sudo ./5.py demo
The absolute path are :  /home/shuhari/pylabs/demo
The absolute path are :  /home/shuhari/pylabs/demo
The absolute path are :  /home/shuhari/pylabs/demo
The absolute path are :  /home/shuhari/pylabs/demo
shuhari@debian:~/pylabs$
```