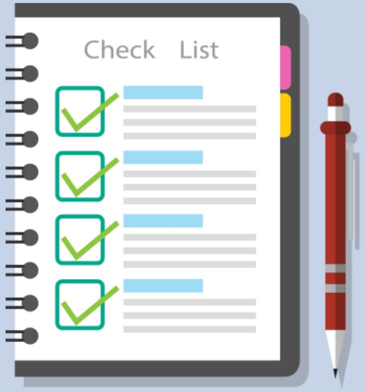
Python – Basics

Digital Lync

INNOVATION - EDUCATION - INCUBATION

Shah Ayub Quadri
aquadri@digital-lync.com

Index

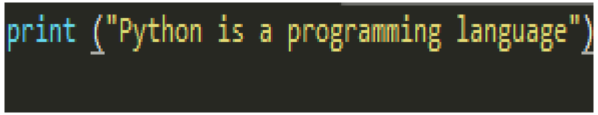


File operations

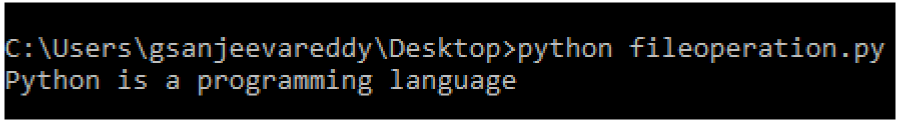
Printing to screen:

•The simplest way to produce output is using the **print** statement where you can pass zero or more expressions separated by commas.

• This function converts the expressions you pass into a string.



```
print ("Python is a programming language")
```



```
C:\Users\gsanjeevareddy\Desktop>python fileoperation.py
Python is a programming language
```

Reading from keyboard:

• Python provides two built-in functions to read a line of text from standard input, which by default comes from the keyboard

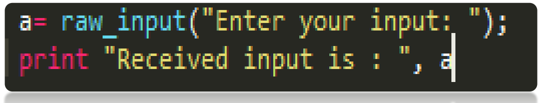
1)raw_input

2)input

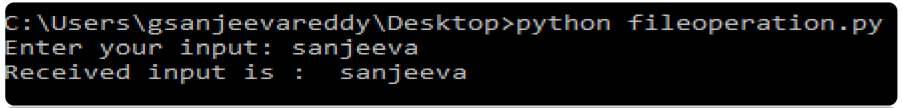
File operations

raw_input **Function:**

» The raw_input([prompt]) function reads one line from standard input and returns it as a string.



```
a= raw_input("Enter your input: ");
print "Received input is : ", a
```

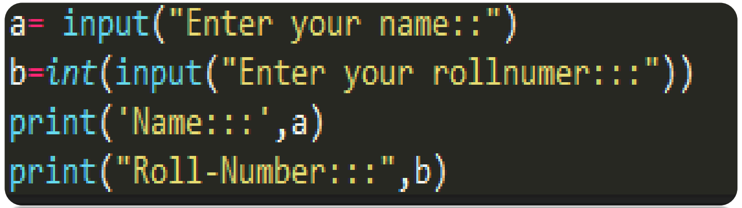


```
C:\Users\gsanjeevareddy\Desktop>python fileoperation.py
Enter your input: sanjeeva
Received input is :  sanjeeva
```

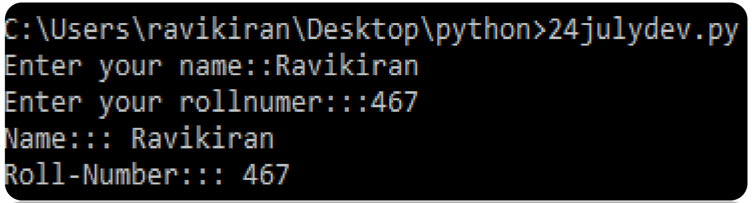
» It will ask prompt you to enter any string and it will display the string which you has enter on the screen.

input Function:

» The *input([prompt])* function is equivalent to raw_input, except that it assumes the input is a valid Python expression and returns the evaluated result to you.



```
a= input("Enter your name::")
b=int(input("Enter your rollnumer:::"))
print('Name:::',a)
print("Roll-Number:::",b)
```



```
C:\Users\ravikiran\Desktop\python>24julydev.py
Enter your name::Ravikiran
Enter your rollnumer:::467
Name::: Ravikiran
Roll-Number::: 467
```

File operations

Opening a file:

» To open a file we have a built-in function called open().

» This function creates a **file** object, which would be utilized to call other support methods associated with it.

Syntax:

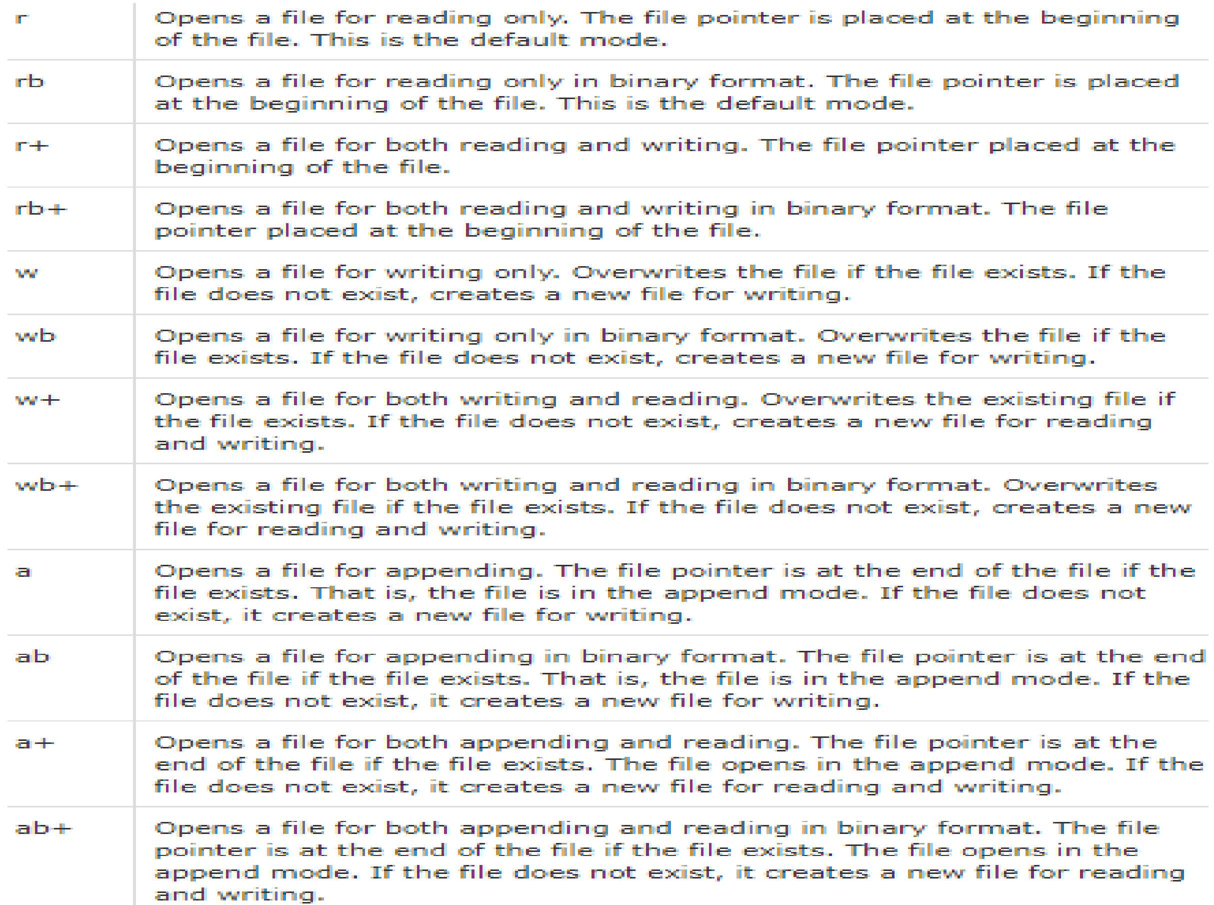
```
file object = open(file_name [, access_mode][, buffering])
```

» **file_name:** The file_name argument is a string value that contains the name of the file that you want to access.

» **access_mode:** The access_mode determines the mode in which the file has to be opened.

» **buffering:** If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file.

File operations

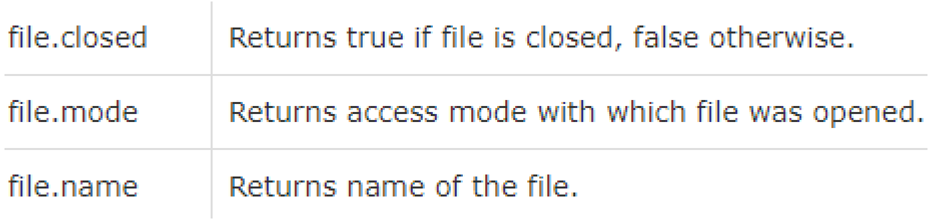


r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

File operations

file Object Attributes:

» After opening a file we have file object with which we can get information about file.



file.closed	Returns true if file is closed, false otherwise.
file.mode	Returns access mode with which file was opened.
file.name	Returns name of the file.

close() Method:

» The close() of a file object is used to close a file after which no write operation is going to be performed.

Syntax:
» fileObject.close();

Example:

```
str = open(“string.txt”,”wb”)
print str.name
str.close()
```

File operations

close():

» If an exception occurs when we are performing some operation with the file, the code exits without closing the file.

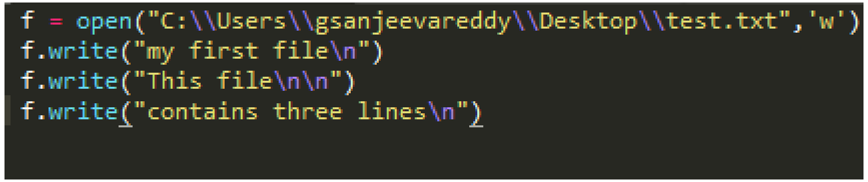
» The safe method is using **try and finally** block.

» This will gurantee that file is closed even if exception is raised.

Write () Method:

» To write into a file in Python, we need to open it in write 'w', append 'a' or exclusive creation 'x' mode.

» 'w' mode as it will overwrite into the file if it already exists.



```
f = open("C:\\Users\\gsanjeevareddy\\Desktop\\test.txt",'w')
f.write("my first file\n")
f.write("This file\n\n")
f.write("contains three lines\n")
```

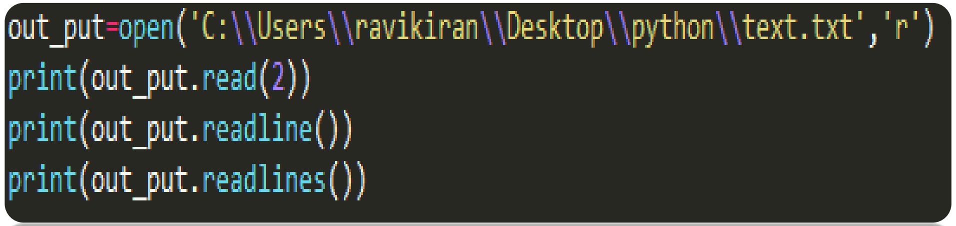
» If the file does not exist it will create a file otherwise it will overwrite the content in file.

» Append will add content to the file with previous content included.

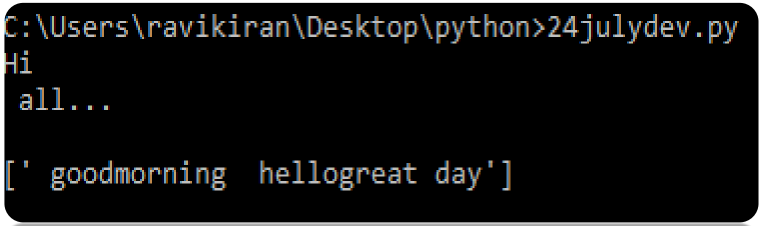
File operations

Reading from file :

- To read a file in Python, we must open the file in reading mode.
- **read(size)** method is used to read in size number of data.
- If **size** is not specified then it will read entire data in file.



```
out_put=open('C:\\Users\\ravikiran\\Desktop\\python\\text.txt','r')
print(out_put.read(2))
print(out_put.readline())
print(out_put.readlines())
```



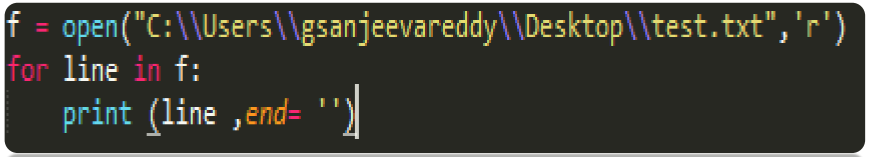
```
C:\Users\ravikiran\Desktop\python>24julydev.py
Hi
 all...

[' goodmorning  hellogreat day']
```

- We can see here with size specified it will read till size of data otherwise it is reading entire data in file.
- We can use tell() method for finding current position of the file.   (**f.tell()**)
- We can use seek method for changing the position in the file.   **(f.seek(0))**

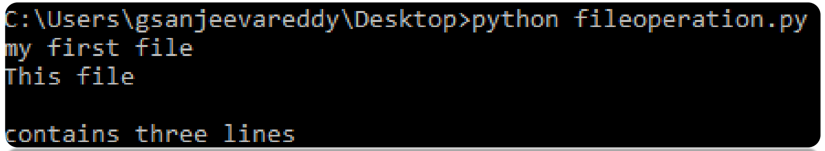
File operations

Reading from file:

• To read a file line by line we can use **for loop.** This the best and efficient way.



```
f = open("C:\\Users\\gsanjeevareddy\\Desktop\\test.txt",'r')
for line in f:
    print (line ,end= '')
```

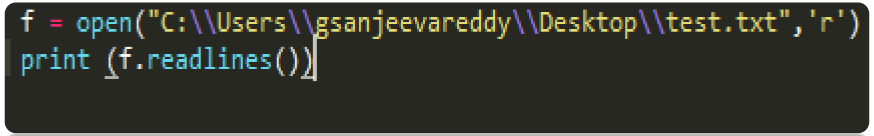


```
C:\Users\gsanjeevareddy\Desktop>python fileoperation.py
my first file
This file

contains three lines
```

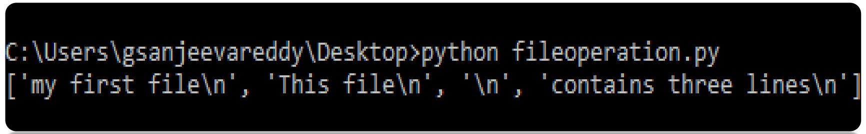
• print() end parameter to avoid two newlines when printing.

• **readline()** method to read individual lines of a file. This method reads a file till the newline, including the newline character.

• **readlines()** method returns a list of remaining lines of the entire file.



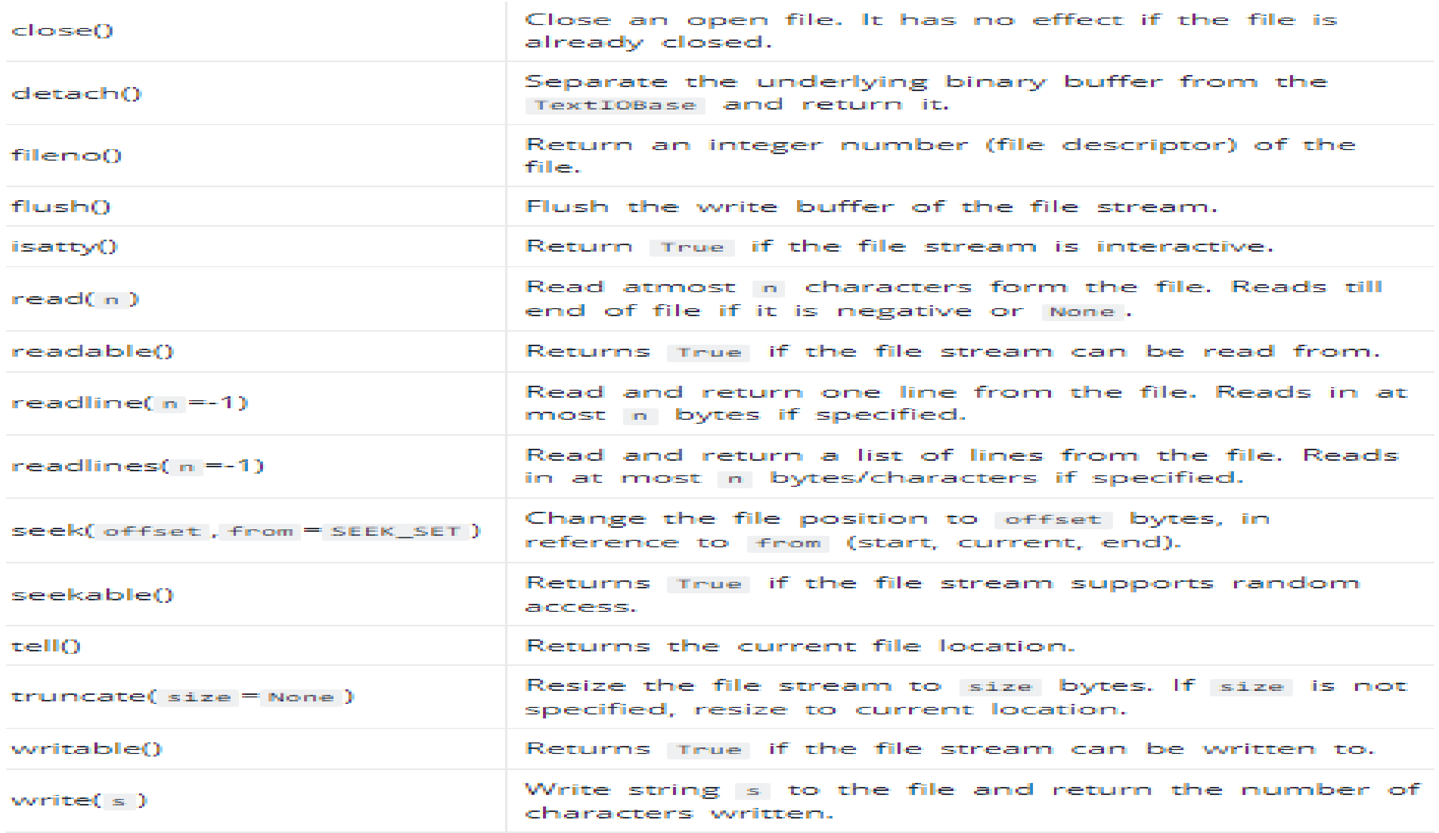
```
f = open("C:\\Users\\gsanjeevareddy\\Desktop\\test.txt",'r')
print (f.readlines())
```



```
C:\Users\gsanjeevareddy\Desktop>python fileoperation.py
['my first file\n', 'This file\n', '\n', 'contains three lines\n']
```

File operations

File Methods:



close()	Close an open file. It has no effect if the file is already closed.
detach()	Separate the underlying binary buffer from the `TextIOBase` and return it.
fileno()	Return an integer number (file descriptor) of the file.
flush()	Flush the write buffer of the file stream.
isatty()	Return `True` if the file stream is interactive.
read(`n`)	Read atmost `n` characters form the file. Reads till end of file if it is negative or `None`.
readable()	Returns `True` if the file stream can be read from.
readline(`n` =-1)	Read and return one line from the file. Reads in at most `n` bytes if specified.
readlines(`n` =-1)	Read and return a list of lines from the file. Reads in at most `n` bytes/characters if specified.
seek(`offset` , `from` = `SEEK_SET`)	Change the file position to `offset` bytes, in reference to `from` (start, current, end).
seekable()	Returns `True` if the file stream supports random access.
tell()	Returns the current file location.
truncate(`size` = `None`)	Resize the file stream to `size` bytes. If `size` is not specified, resize to current location.
writable()	Returns `True` if the file stream can be written to.
write(`s`)	Write string `s` to the file and return the number of characters written.

File operations

Renaming file:

- Python os provides file processing methods **renaming** and **deleting.**
- Firstly we need to **import os** and we call related functions.

Rename() method:

- **rename()** method takes two arguments the **current file name** and **new file name.**

Syntax:

```
os.rename(current_file_name, new_file_name)
```

Example:

```
import os
os.rename(“test.txt”, “test1.txt”)
```

File operations

remove():

» *remove()* method to delete files by supplying the name of the file to be deleted as the argument.

Syntax:

```
os.remove(file_name)
```

Example:

```
import os
os.remove(“test1.txt”)
```

File operations

getcwd() method:

- It will display the current working directory location.

Syntax:

```
os.getcwd()
```

Example:

```
import os
os.getcwd()
```

rmdir() method:

- rmdir() will delete the directory that is specified.

Syntax:

```
os.rmdir(“dirname”)
```

Example:

```
import os
os.rmdir(“first/team”)
```

- Here we have to specify the full qualified name of the directory.
- Other wise it will search in current working directory

Assignment

1. Write a Python program to read an entire text file.

2. Write a Python program to read first n lines of a file.

3. Write a Python program to append text to a file and display the text.

4. Write a Python program to read last n lines of a file.

5. Write a Python program to read a file line by line and store it into a list.