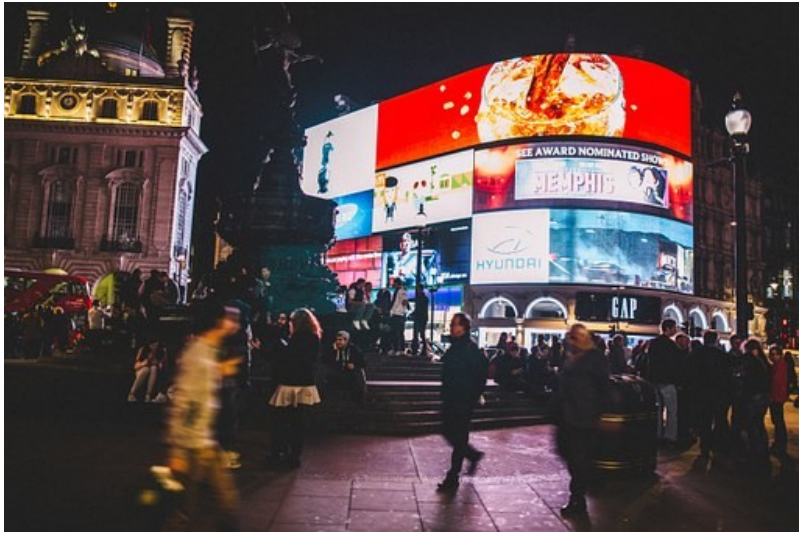
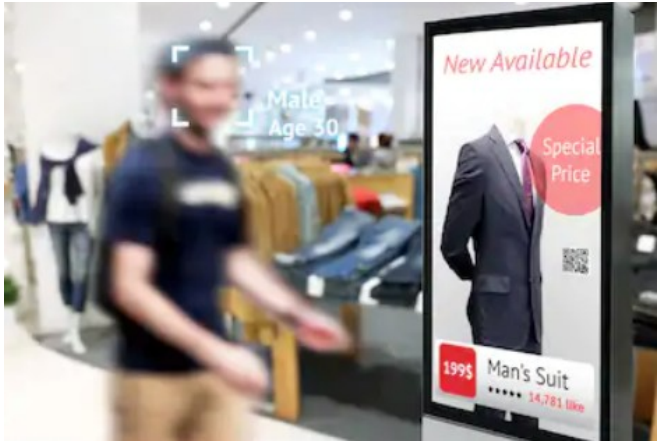

∇ueron.△i

DOOH(Digital Out of Home) Advertising

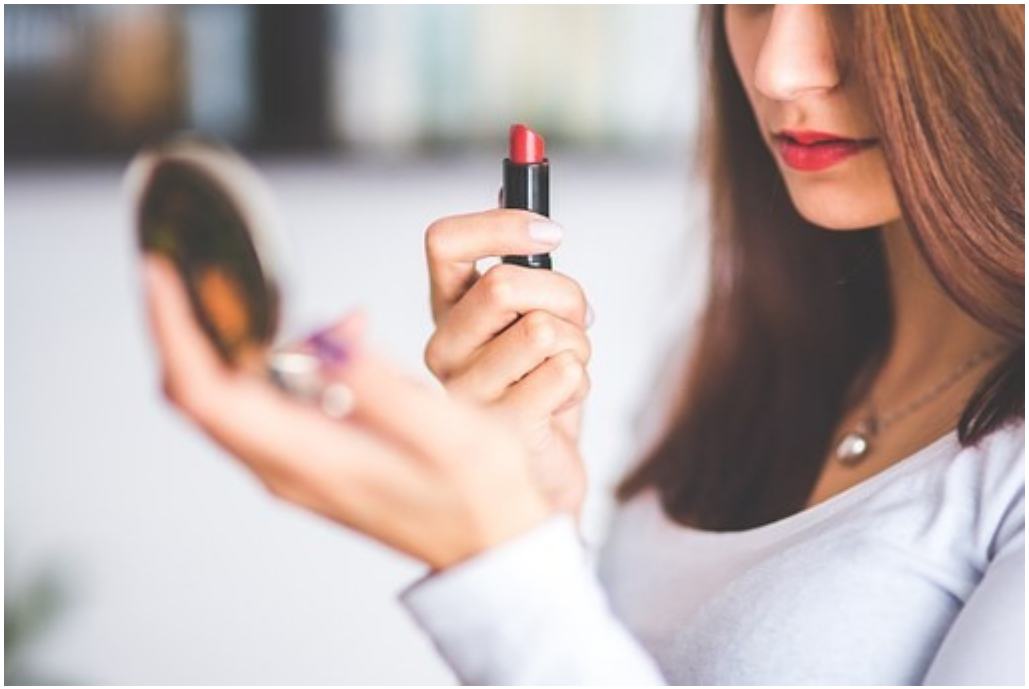




VUERON is working to:

- Bring about an innovative and cost-effective digital system of advisement flow.
- Provide an easily accessible platform for masses.
- Bridge the gaps in the chain between advertisers and buyers.
- Overcome the barriers of conventional marketing.
- Empower advertisers to reach their exact audiences in a better manner.

Example: Ad of a Lipstick



Any guesses where do we ***jump*** in???

VUERON SYSTEMS PVT. LTD.

	https://www.vueron.ai
	https://www.linkedin.com/company/vueron
	http://www.facebook.com/vueron
	http://www.twitter.com/vueron_ai
	https://instagram.com/vueron_ai

Vueron.Ai

∇ueron.∆i

A three days Workshop

by Ravin Kumar

DAY 1 : PYTHON PROGRAMMING

DAY 2 : AUTOMATION & IMAGE PROCESSING

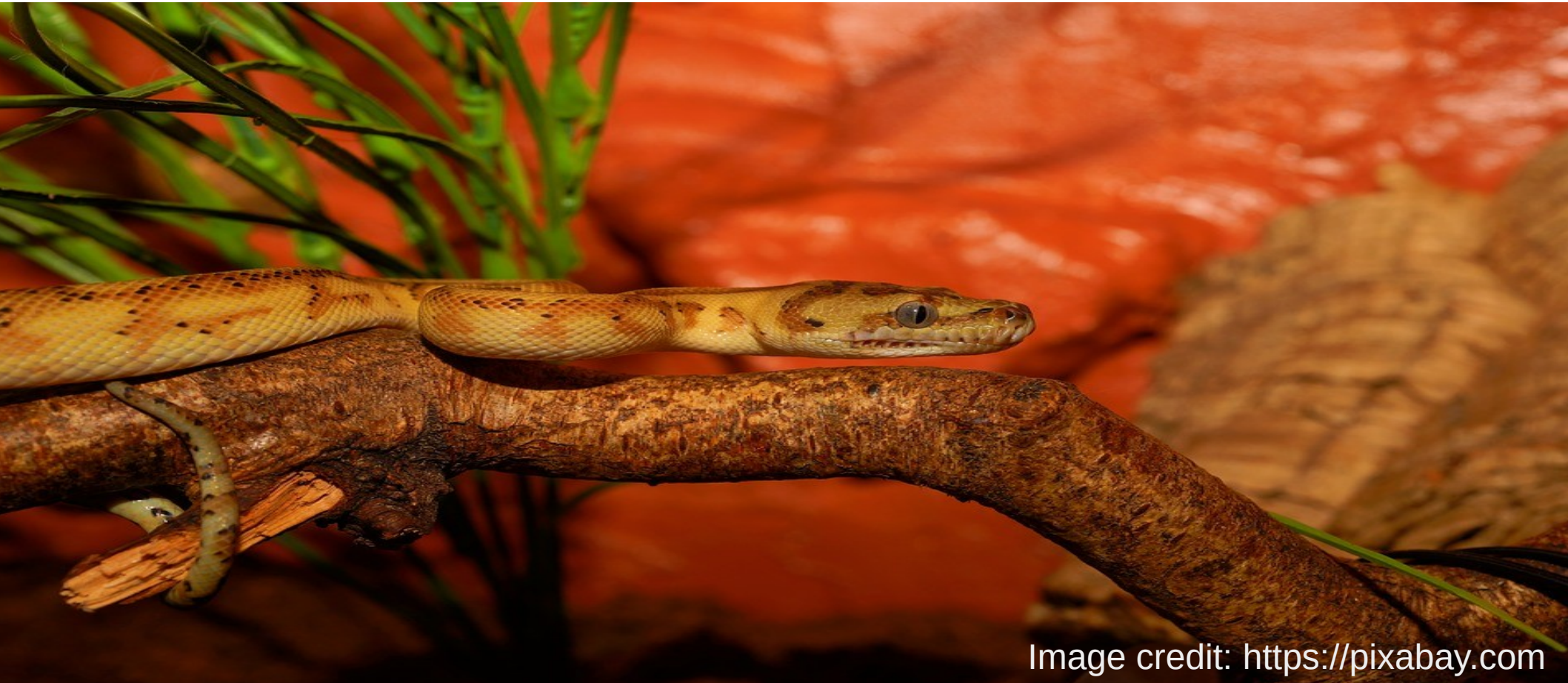
DAY 3 : MACHINE LEARNING

∇ueron.Δi

Python Programming

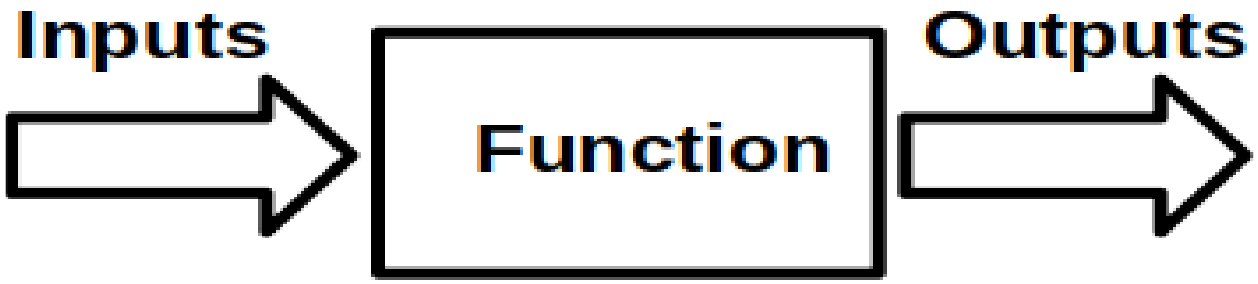
Presentation Link: https://github.com/mr-ravin/python3-workshop

What is Python ?



https://github.com/mr-ravin/python3-workshop

Let’s do some mathematics



Let’s do some mathematics

Consider a function, f(x) = x+2

when x = 1, f(1)= 3

x = 3, f(3)= 5

x = 5, f(5)= 7

How can we implement this function in programming ?

Basics of a function

In computer science, a function can be of many types

S.NO	Input Parameter	Output Return
1	NO	VOID
2	NO	YES
3	YES	VOID
4	YES	YES

Syntax for a function

```
>>> def f( ):     # no input parameter
        -----
        -----      #  no return output

>>> def f( x ):    # input 'x'
        -----
        -----      # no return output
```

Syntax for a function

```
>>> def f(  ):     # no input parameter
        -----
        return res      #  return 'res'

>>> def f( x ):    # input 'x'
        -----
        return res      # return 'res'
```

Numbers

```
>>> value_1 = 11   # this is an integer
>>> value_2 = 7    # this is an integer
>>> value_3 = 1.0  # this is a float
>>> value_4 = 1/2  ## this is an NOT integer
                   ## in Python 3.6, But is an
                   ## integer in Python 2.7
```

Numbers

```
>>> value_5 = 1.0 / 2    # this is a float
>>> value_6 =  1 / 2.0   # this is a float
>>> value_7 =  1.0 / 2.0 # this is a float
>>> value_8 = 1.0 * 1    # this is a float
>>> value_9 = 1.0 + 1    # this is a float
```

What about these ?

```
>>>  value_10 = 1.0 – 1 # this is ?
>>>  value_11 = 1.0 * 0 # this is ?
```

Convert int to float

```
>>> value_12 = 21  # integer
>>> value_12_new = float( value_12 )
>>> type( value_12 )
>>> type( value_12_new )
```

Convert float to int

```
>>> value_13 = 12.0   # float
>>> value_13_new =  int( value_13 )
>>> type( value_13 )
>>> type( value_13_new )
```

Numbers, and Lists

```
>>> this_is_an_integer = 1
>>> this_is_a_float = 1.0
>>> list_of_integers= [ 1, 2, 3 ]
>>> list_of_floats=[ 1.0, 2.0, 3.0 ]
>>> list_of_numbers=[ 1, 2.0, 3/2 ]
```

Indexing in Lists

```
>>> a = [ 21, 22, 23, 24 ] # index starts with 0
>>> print( a ) # this function prints entire list a
>>> print( a[0] ) # element at zero index i.e 21
>>> print( a[1] ) # element at 1 index i.e. 22
>>> print( a[-1] ) # element at last index i.e. 24
```

List Operations

```
>>> a = [ 0 ]
>>> a = a.append(1)
>>> print(a)  # value of updated list ?
>>> b = [2, 3, 4]
>>> c = a.extend(b)
>>> print(c) # value of updated list ?
```

List of Lists

```
>>> list_a = [ 1, 2, 3.0 ]
>>> list_b = [ 7, 8.0, 0 ]
>>> list_a[ 0 ] = list_b        # what will happen ?
>>> print( list_a )
>>> print( list_a[0][0] )       # what will happen ?
>>> print( list_a[0][-1] )      # what will happen ?
```

Copying a List

```
>>> list_1 = [ 1, 2 ,3 ]
>>> list_2 = [ 7, 8 ,9 ]
>>> list_1[0] = list_2
>>> list_2[0] = -1
>>> print( list_1 )
```

Copying a List

```
>>> list_1 = [ 1, 2 ,3 ]
>>> list_2 = [ 7, 8 ,9 ]
>>> list_1[0] =list_2.copy( )      # copying the list
>>> list_2[0] = -1
>>> print( list_1 )
>>> print( list_2 )
```

Indexing in depth

```
>>> list_a = [ 91, 92, 93, 94, 95 ]
>>> new_list = list_a[ 1 : 3 ]  ## represent [ 1, 3)
                                ##     index values
>>> print( new_list )           # what will happen ?
>>> print( list_a )             # what will happen ?
>>> print( list_a[3:-1])        # what will happen ?
```

Copying a List

```
>>> list_1 = [ 1, 2 ,3 ]
>>> list_2 = [ 7, 8 ,9 ]
>>> list_1[0] =list_2[ : ]      # copying the list
>>> list_2[0] = -1
>>> print( list_1 )
>>> print( list_2 )
```

Length Function

```
>>> list_1 = [1, 2, 3, 4, 50 ]
>>> print (   len(  list_1  )    )          ## prints 5 as total no. Of
                                             ## elements in list_1 is 5

>>> list_2 = [ 9, 8 ]
>>> print (   len(  list_2  )    )
>>> list_3 = list_2.copy( )
>>> list_3[0] = list_1
>>> print ( len( list_3 ) )                  # what will happen ?
```

Range Function

```
>>> range_val_1 = range( 5 )  #  [ 0, 1, 2, 3, 4 ]
>>> range_val_2 = range ( 2, 5 ) # [ 2, 3, 4 ]
```

```
print("break time")
```

https://github.com/mr-ravin/python3-workshop

Swapping

Let us say **a = 2**, and **b =7**, now swap(i.e interchange) the values of ***a*** and ***b***

```
>>>  a = 2
>>>  b = 7
>>>  c = a
>>>  a =  b
>>>  b = c
>>>  print(a)
>>>  print(b)
```

Relational Operators

S.NO	RELATION	SYMBOL
1	GREATER THAN	>
2	GREATER THAN OR EQUAL	> =
3	LESS THAN	<
4	LESS THAN OR EQUAL	< =
5	EQUAL TO	==
6	NOT	!
7	NOT EQUAL	! =

if-elif-else

```
value_a = 10
value_b = 7
if value_a > value_b:
    print(“value_a is greater than value_b”)
print(“completed”)
```

if-elif-else

```
value_a = 10
value_b = 7
if value_a > value_b:
    print(“value_a is greater than value_b”)
else:
    print(“value_a is less than or equal to value_b”)
print(“completed”)
```

if-elif-else

```
value_a = 10
value_b = 7
if value_a > value_b:
    print(“value_a is greater than value_b”)
elif value_a == value_b:
    print(“value_a is equal to value_b”)
else:
    print(“value_a is less than to value_b”)
print(“completed”)
```

https://github.com/mr-ravin/python3-workshop

Loops

1. **for** loop (element based)
2. **while** loop (condition based)

for loop

```
>>> for  elem in range(0,10):
        print(elem)
```

while loop

```
>>> a = 12
>>> b = 7
>>> while a > b :
        print(“a is greater than b”)
        a = a - 1
```

Sum of ***N*** natural numbers with loops

```
>>> sum = 0
>>> n = int( input(  ) )
>>> for elem in range( n+1 ):
        sum = sum + elem
>>> print(sum)
```

Sum of *N* natural numbers with loops

```
>>> cnt = 1
>>> sum = 0
>>> n = int( input(  ) )
>>> while cnt < = n:
        sum = sum + cnt
        cnt = cnt + 1
>>> print(sum)
```

Strings

```
>>> s_1 = “this is one string”
>>> s_2 = “this is also 1 string”
>>> s_3 = “this is ” + ”also 1 string”
>>> s_4 = “this is also ”+ str(1) + ” string”
```

Operation on Strings

```
>>> s_1 = “this is a string”
>>> length_of_s1 = len(s_1)
>>> print(length_of_s1)
>>> s_2 = s_1[ 1:4 ]
>>> print(s_2)
```

Split and Join

```
>>>  s_data = “abc @company.com”
>>>  s_split = s_data.split(“@”)
>>>  print(s_split)
```

Split and Join

```
>>>  s_data = “abc @company.com”
>>>  s_split = s_data.split(“@”)
>>>  s_data = “@”.join(s_split)
>>> print(s_data)
```

Tuples & Dictionary

```
>>> t1 =( 7,0,11 ) ## once defined, we can not
                   ##change length of a tuple
>>> print(t1)
>>> print(t1[0] )
```

Tuples

```
>>> t1 = ( 1, 2, 3 )
>>> list_a = [ 9, 10, 11 ]
>>> t1[0] = list_a.copy( ) # Error
>>> t1 = ( [1], 2, 3 )
>>> t1[0] = t1[0].append( 9 ) # it is possible !!!
>>> print( t1 )
```

Dictionary { }

```
>>> dict_a = { “fruit”: “apple”, “cards”: [ 1, 2, 3 ] }
>>> print( dict_a[“fruit”] )
>>> dict_a[ “cards” ] = [ 4, 5, 6, 7 ]
>>> print( dict_a[“cards”] )
>>> dict_a[ “cards” ].append( 10 )
>>> print( dict_a[“cards”] )
```

Functions without parameters

```
>>> def square_1( ):
        x = int(input( ))
        print( x**2 )

>>> def square_2( ):
        x = int(input( ))
        return x**2
```

Functions without parameters

```
>>> square_1( )
>>> square_2( )  #  store the return value ?


>>> result = square_2( )
>>> print( result )
```

Functions with parameters

```
>>> def square_3( x ):
        print( x**2 )

>>> def square_4(x ):
        return x**2
```

Functions with parameters

```
>>> x = int( input( ) )
>>> square_3( x )


>>> result_4 = square_4( x )
>>> print( result )
```

Lets make this function

Write a function **f** such that-

f(x**)** = x , when x >0

and, **f(**x**)** = 0, when x < = 0

Surprise !!!

```
>>> def f(x):
        if x < = 0:
            x = 0
        return x
```

Recursion

Question: Let us assume f(x) = x+ 2, now calculate f(f(x)) for x = 7, 9, and 11.

Recursion

Question: Let us assume f(x) = x+ 2, now calculate f(f(x)) for x = 7, 9, and 11.

1. f(f(7)) = 11
2. f(f(9)) = 13
3. f(f(11)) = 15

Sum of *N* natural numbers using recursion

```
>>> def sum_n(n,s):
        if n == 0:
            return s
        else:
            s = s + n
            n = n - 1
            s = sum_n(n,s)
            return s
```

Iterations

- In this, a **small** step is **repeated** for large number of times to reach the goal.

Sum of *N* natural numbers using iteration

```
>>> def sum_n(n):
        cnt = 0
        s = 0
        while cnt < =n:
            s = s + cnt
            cnt = cnt + 1
        return s
```

Importing / using standard **Libraries**

```
>>> import os
>>> os.system(“echo \“ a new file \” >> a.txt ”)


>>> import datetime
>>> data = datetime.datetime.now()
>>> print(data)
```

https://github.com/mr-ravin/python3-workshop

Command Line Arguments

It allows passing details to a program before it is **run.**

It is helpful for a program to know some details ahead of its running. **Example- operating system version, hardware type.**

Single command-line argument

Consider following source code of a python3 file

abc.py

```
import sys
arg1 = sys.argv[1]
print(arg1)
```

run command: python3 abc.py **thisdata**

Multi command-line arguments

Consider following source code of a python3 file

abc.py

```
import sys
arg1 = sys.argv[1]
arg2 = sys.argv[2]
print(arg1)
print(arg2)
```

run command: python3 abc.py **this data**

Mathematical Operations

```
>>> import math
>>> var_1 = 16
>>> var_2 =  math.sqrt( var_1)
>>> print(var_2)
>>> val_pi = math.pi       # Pi
>>> print(var_pi)
```

Mathematical Operations

```
>>> import math
>>> val_1 = math.sin(0)
>>> val_2 = math.sin(90)
>>> print(val_1)
>>> print(val_2)      # is val_2 == 0 ?
```

Mathematical Operations

```
>>> import math
>>> val_1 = 64
>>> base_value = 2
>>> val_2 =  math.log(val_1, base_value)
```

File Operations

Read a file:

```
>>> file_name= “sample.txt”
>>> data_link=open(file_name, “r”)
>>>  data=data_link.readlines()
>>>  print(data)
>>>  data_link.close()
```

File Operations

Write a file:

```
>>> file_name= “sample.txt”
>>> data_to_write=”this is the data”
>>> data_link=open(file_name, “w”)
>>>  data=data_link.write(data_to_write) ##   pass string
                                          ##  type data
>>>  data_link.close()
```

File Operations

Append a file:

```
>>> file_name= “sample.txt”
>>> data_to_write=”this is the data”
>>> data_link=open(file_name, “a”)
>>>  data=data_link.write(data_to_write) ##   pass string
                                          ##  type data
>>>  data_link.close()
```

Other methods to read a file

```
>>> data_link = open(“sample.txt”, “r”)
>>>
>>>data = data_link.read()
```

OR

```
>>> data = data_link.readline()
```

OR

```
>>>  data = data_link.readlines()
>>>
>>> data_link.close()
```

print(“Discussion”)

https://github.com/mr-ravin/python3-workshop

Presentation Link:

https://github.com/mr-ravin/python3-workshop