CSB101: Problem Solving And Computer Programming Lab

LAB ASSIGNMENT 7: Function in C

Date Assigned : 7/8/2023 **Date Submitted : 13/8/2023**

Submitted By:

Name: Rohit Kumar

Roll No: 211210053

Branch: CSE

Semester: 1st Sem

Group : 2

Submitted To: Dr. Chandra Prakash

Department of Computer Science and Engineering

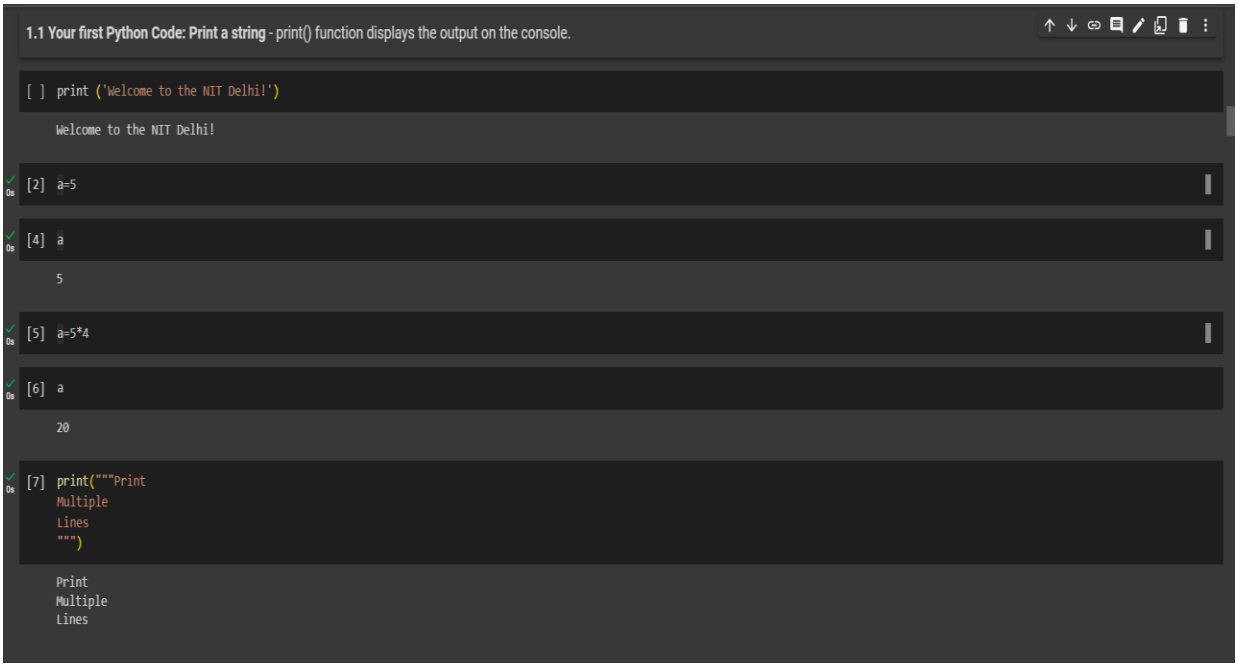
NATIONAL INSTITUTE OF TECHNOLOGY DELHI

2021

PART A : Conceptual Questions

1.1 Your first Python Code: Print a string - print() function displays the output on the console.

OUTPUT:



1.1 Your first Python Code: Print a string - print() function displays the output on the console.

```
[ ] print ('Welcome to the NIT Delhi!')
```

```
Welcome to the NIT Delhi!
```

```
[2] a=5
```

```
[4] a
```

```
5
```

```
[5] a=5*4
```

```
[6] a
```

```
20
```

```
[7] print("""Print
    Multiple
    Lines
    """)
```

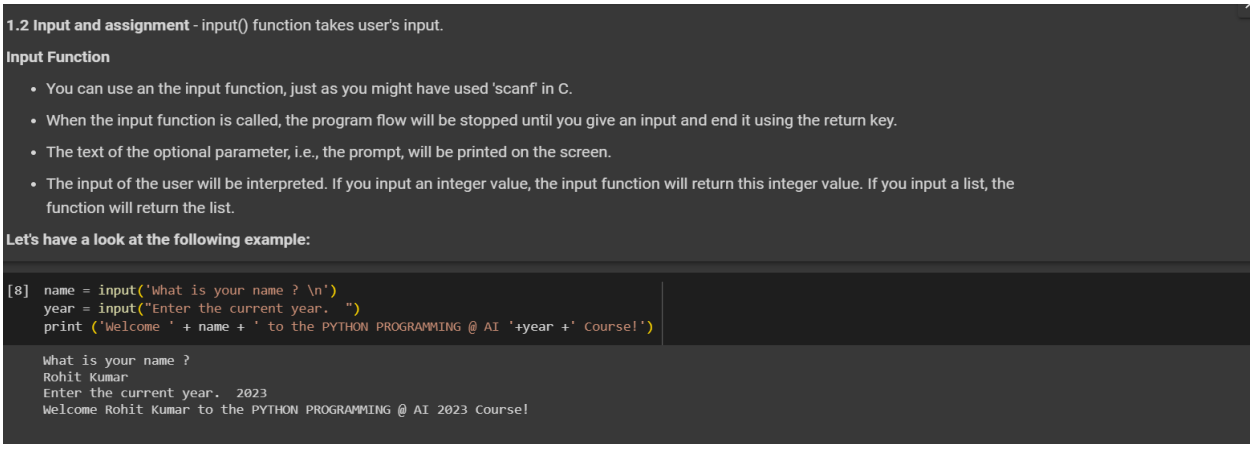
```
Print
Multiple
Lines
```

OBSERVATION

Single line, multiline and variables can be printed with printed with the help of print() function.

1.2 Input and assignment - input() function takes user's input.

OUTPUT:



1.2 Input and assignment - input() function takes user's input.

Input Function

- You can use an the input function, just as you might have used 'scanf' in C.
- When the input function is called, the program flow will be stopped until you give an input and end it using the return key.
- The text of the optional parameter, i.e., the prompt, will be printed on the screen.
- The input of the user will be interpreted. If you input an integer value, the input function will return this integer value. If you input a list, the function will return the list.

Let's have a look at the following example:

```
[8] name = input('What is your name ? \n')
    year = input("Enter the current year.  ")
    print ('Welcome ' + name + ' to the PYTHON PROGRAMMING @ AI '+year +' Course!')
```

```
What is your name ?
Rohit Kumar
Enter the current year.  2023
Welcome Rohit Kumar to the PYTHON PROGRAMMING @ AI 2023 Course!
```

```
for number in [1,10,20,30,40,50]:
    print(number,"\n")

# Identify the Error and resolve
```

```
1

10

20

30

40

50
```

- As you can see above it is giving IndentationError
- We can overcome this error by providing a single/multiple space or tab seperation

```
for number in [1,10,20,30,40,50]:
    print(number,"\n")
```

```
1

10

20

30
```

OBSERVATION

A variable type is not needed while declaring the variable. Input() function can be used to get string and integer input. Concatenation of string can be done using '+' operator.

Indentation is important and python and python is case sensitive.

1.3 If Else Statement -

OUTPUT

```
num = input("number :")
if int(num) > 0:
    print("Positive number")
elif int(num) == 0:
    print("Zero")
else:
    print("Negative number")
```

```
number :-21
Negative number
```

OBSERVATION

The conditional statements are navigated through if,else,elif.

1.4 Loops in Python - We can iterate loop using various types. Here we are using built-in enumerate function.

OUTPUT

OBSERVATION

List can be traversed using for, in and enumerate(in-built function).

1.5 While Loop - Same as other programming languages

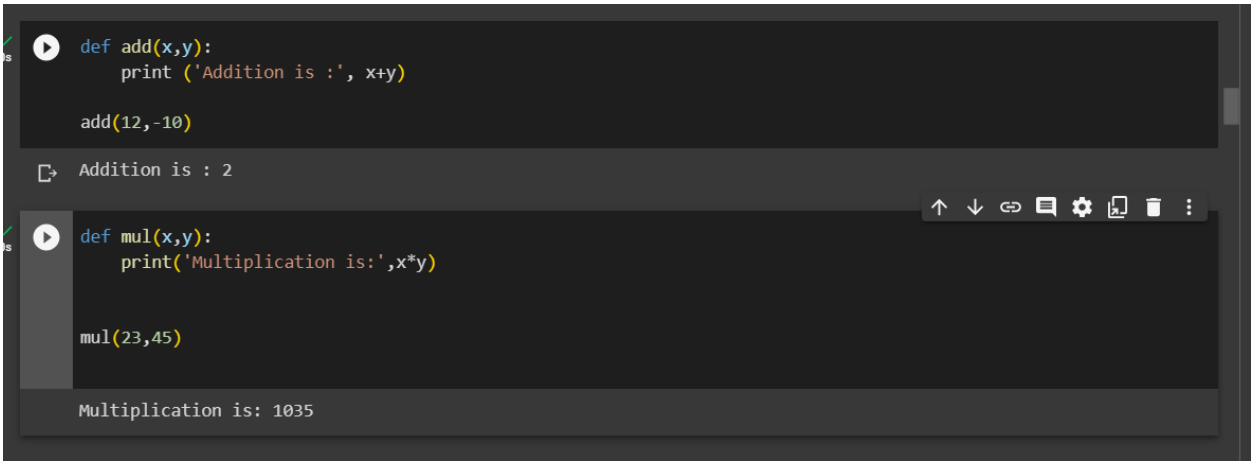
OUTPUT

OBSERVATION

Lists can also be traversed through while loop aswell, len() is used get the size of a list.

1.6. Defining Function - Just write def function_name(parameters)

OUTPUT



```
def add(x,y):
    print ('Addition is :', x+y)

add(12,-10)
```

```
Addition is : 2
```

```
def mul(x,y):
    print('Multiplication is:',x*y)


mul(23,45)
```

```
Multiplication is: 1035
```

OBSERVATION

def function_name(parameters) is used to define a function and function_name() can be used to call that function. The indentation of function and calling is same.

1.7 Dictionary - Dictionary is an associative array of python

OUTPUT

```
prices = {'apple': 100, 'banana': 50}
my_purchase = {'apple': input('apple :'),'banana': input('banana :')}
grocery_bill = sum(prices[fruit] * int(my_purchase[fruit])
                   for fruit in my_purchase)
print ('Total Bill : %.2f' % grocery_bill +' rupees')
```

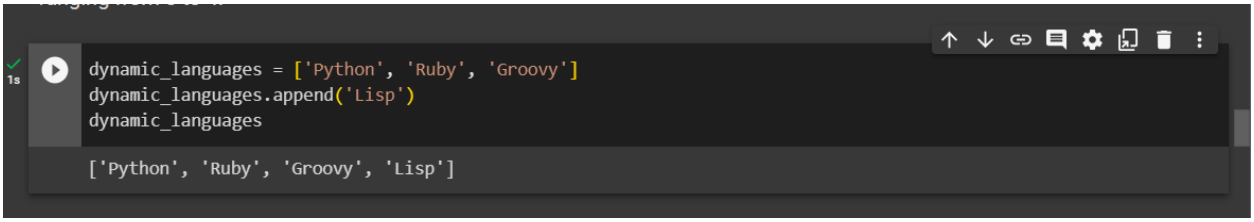
```
apple :5
banana :6
Total Bill : 800.00 rupees
```

OBSERVATION

Dictionaries are key value pairs, sort of like maps and they can be accessed through index or key values.

1.8 List –

OUTPUT



```
dynamic_languages = ['Python', 'Ruby', 'Groovy']
dynamic_languages.append('Lisp')
dynamic_languages
```

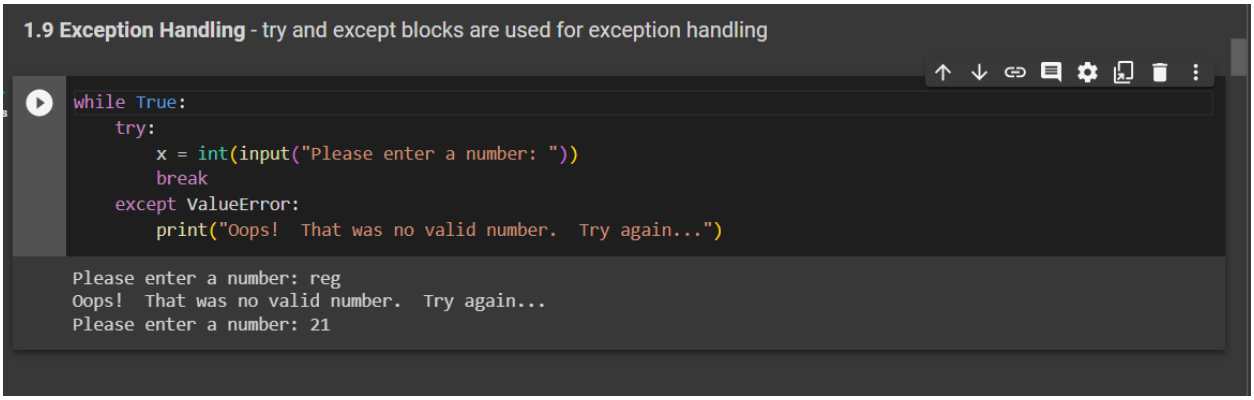
```
['Python', 'Ruby', 'Groovy', 'Lisp']
```

OBSERVATION

.append() is an in-built function through which we can add elements to a list.

1.9 Exception Handling - try and except blocks are used for exception handling

OUTPUT



1.9 Exception Handling - try and except blocks are used for exception handling

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")
```

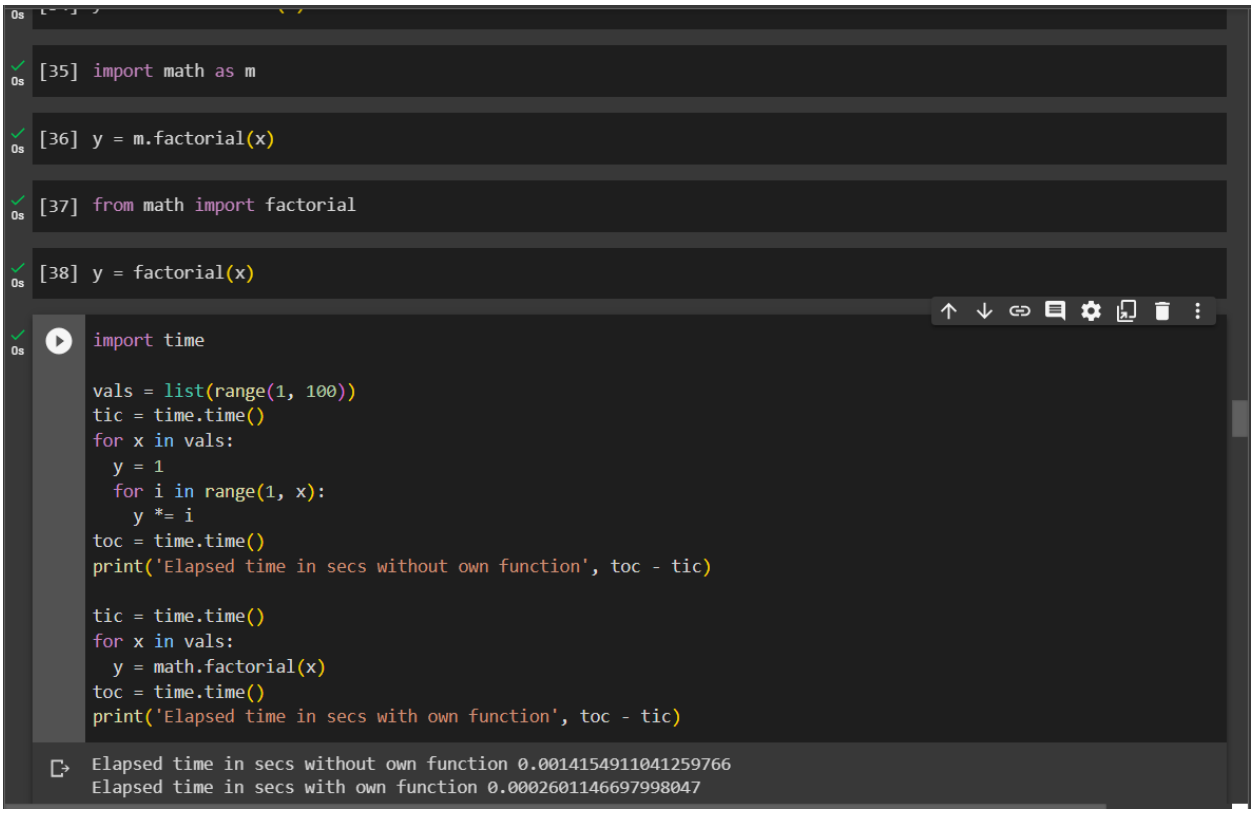
```
Please enter a number: reg
Oops!  That was no valid number.  Try again...
Please enter a number: 21
```

OBSERVATION

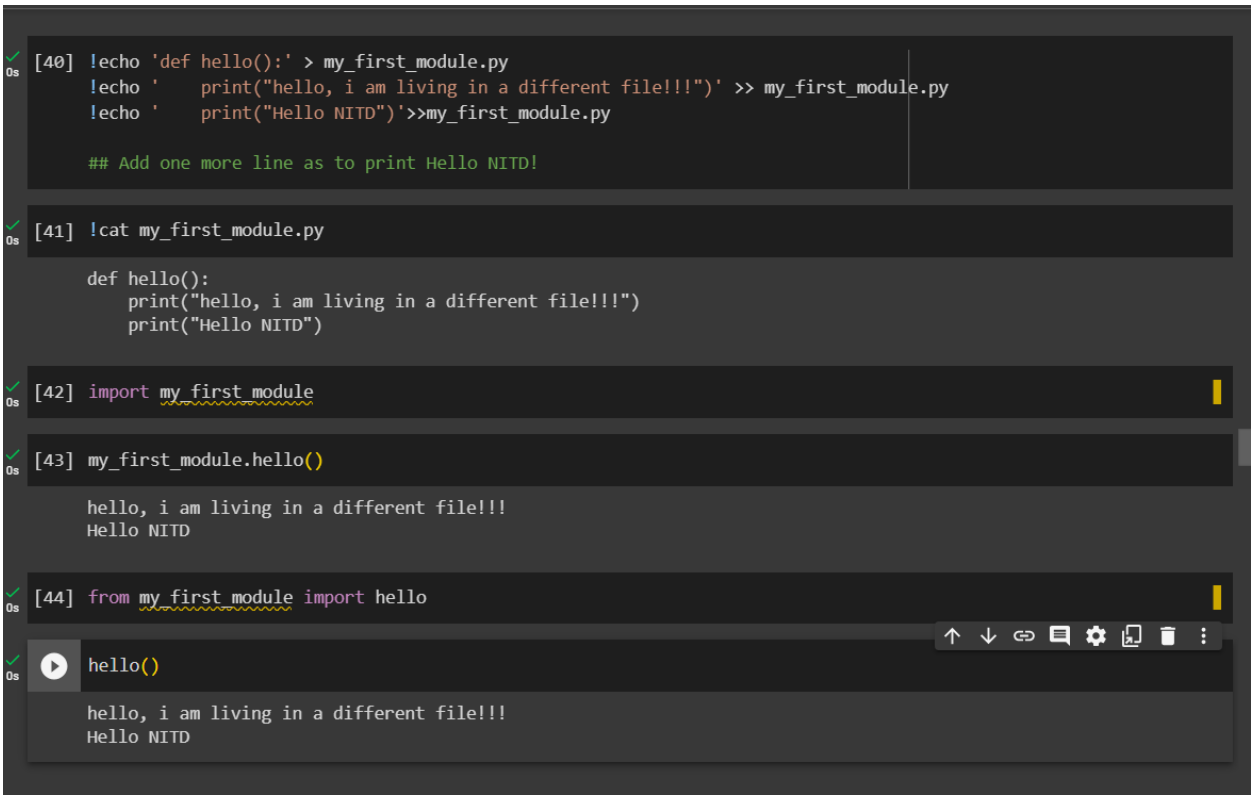
Try except block works exactly as try try,throw catch block in java or C++. In try block , the condition is checked if the condition is not met, except is executed.

Python Packages.

```
Python Packages

[28] import math

[29] math?

[30] a = math.sqrt(100)
     print(a)

     10.0

[31] a = math.pow(100, 0.5)
     print(a)

     10.0

     x = 100
     y = 1
     for i in range(1, x):
       y *= i
     print('Factorial of', x, 'is', y)

     Factorial of 100 is 933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761

[ ]  y = math.factorial(x)
```



```
[35] import math as m

[36] y = m.factorial(x)

[37] from math import factorial

[38] y = factorial(x)

     import time

     vals = list(range(1, 100))
     tic = time.time()
     for x in vals:
       y = 1
       for i in range(1, x):
         y *= i
     toc = time.time()
     print('Elapsed time in secs without own function', toc - tic)

     tic = time.time()
     for x in vals:
       y = math.factorial(x)
     toc = time.time()
     print('Elapsed time in secs with own function', toc - tic)

     Elapsed time in secs without own function 0.0014154911041259766
     Elapsed time in secs with own function 0.0002601146697998047
```



FILE HANDLING

```
with open('mobile_cleaned.csv', 'r') as file:
  print(file.read())
```

```
PhoneId,Pixel Density,Screen Size,Weight,RAM,Processor_frequency,Screen to Body Ratio (calculated),Height,Internal Memory,Capacity,Resolution,SIM 2_2G,SIM 2_3G,SIM 2_4G,SIM 2_Other,
0,403,6.26,182,4,1.8,80.68,157.9,64,4000,20,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0
1,271,6.2,168,3,1.8,80.85,156.2,32,4230,8,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0
2,409,6.3,168,3,2.1,83.68,157.0,32,3500,25,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
4,411,6.0,169,4,2.2,74.78,159.8,64,3300,24,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
5,396,6.5,175,4,2.2,84.23,160.4,64,3750,16,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
7,402,6.41,156,6,2.2,84.98,157.9,64,3400,25,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0
8,295,5.45,145,2,2.0,72.68,147.5,16,3000,5,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0
9,432,5.84,178,3,2.0,79.24,149.3,32,4000,5,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0
10,271,6.2,168,2,1.8,80.85,156.2,16,4230,8,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
12,403,6.18,182,6,2.8,82.2,155.5,64,4000,20,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0
14,402,6.41,185,6,2.8,85.43,157.5,128,3700,16,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0
15,403,5.99,168,4,2.2,77.38,158.7,64,3010,20,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0
17,432,5.84,152,3,2.36,79.07,149.2,32,3000,16,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0
18,400,6.3,150,4,2.2,84.87,154.81,64,3260,24,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0
20,270,6.22,161,4,2.0,82.6,155.1,32,4030,8,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0
21,435,5.8,151,4,1.8,80.05,147.2,64,3060,16,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
22,403,6.18,180,4,2.2,82.03,154.8,64,3500,20,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0
23,270,6.22,161,4,1.95,82.6,155.1,64,4030,20,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0
26,409,6.3,176,4,2.36,82.98,157.9,64,3750,16,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0
28,269,6.26,160,3,1.8,80.64,158.4,32,4000,8,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
30,271,6.2,168,4,1.8,81.22,155.9,64,4230,16,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
31,403,6.26,182,6,1.8,80.68,157.9,64,4000,20,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0
33,270,6.22,143,2,2.0,82.66,155.0,16,3260,5,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0
34,287,5.86,151,3,1.8,79.34,149.5,32,3060,8,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
36,274,6.0,178,2,1.4,73.94,161.4,32,3300,5,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
37,269,5.99,170,4,2.0,74.64,160.7,64,3080,16,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0
38,392,6.3,183,6,2.2,80.86,162.5,128,3800,24,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0
42,388,6.59,199,8,2.8,86.47,162.0,128,4000,8,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0
44,403,6.26,175,4,1.95,81.64,157.9,64,5000,13,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0
45,538,6.39,189,6,2.6,87.66,157.8,128,4200,24,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0
```

```
with open('mobile_cleaned.csv', 'r') as file:
  for line in file:
    for word in line.split(','):
      print(word)
    print('-------')
```

```
Streaming output truncated to the last 5000 lines.
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

9s completed at 9:23 PM

```
[58] with open('my_first_file_output.txt', 'w') as file:
       file.write('hello world from python code')
```

```
!cat my_first_file_output.txt
```

```
hello world from python code
```

1.10- NumPy

+ Code + Text

```
import numpy as np

b1 = np.array([1,2,3,5,5,8])    #Declaring a NumPy Array
b2 = np.array([4,5,6,7,7,9])

print(b1+b2)

print(b2 * 3)
print("No. of dimensions: ", b1.ndim)  # Rows in array, considered as a matrix.
# Printing shape of array
print("Shape of array: ", b1.shape)  # Dimension

#reshaping an array
r_b1=b1.reshape(2,3)

print("Reshaped array: ", r_b1)
print("Shape of array: ", r_b1.shape)

# Printing size (total number of elements) of array
print("Size of array: ", b1.size) # elements in a row or column elements.

# Printing the datatype of elements in array
print("Array stores elements of type: ", b1.dtype)
```

```
[ 5  7  9 12 12 17]
[12 15 18 21 21 27]
No. of dimensions:  1
Shape of array:  (6,)
Reshaped array:  [[1 2 3]
 [5 5 8]]
Shape of array:  (2, 3)
Size of array:  6
```

+ Code + Text

```
# Creating array from a list with type float
A = np.array([[1, 2, 4], [5, 8, 7]], dtype = 'float')
print("A : ",A)
# Create a 3X4 array with all zeros. Please note, we have used double paranthesis.
B = np.zeros((3, 4))
print("B : ",B)
# Create an arrary of complex numbers
C = np.full((3, 3), 6, dtype = 'complex')
print("C : ",C)
# Create an array with random values
np.random.seed(1) # A seed is set to ensure that the results are consistent if you use this array in future computations also.

D = np.random.randn(2, 2)
print("D : ",D)
E = np.random.random((2, 2))
print("E : ",E)
F = np.random.randint(3, 15, size=(2, 4))
print("F : ",F)
```

```
A :  [[1. 2. 4.]
 [5. 8. 7.]]
B :  [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
C :  [[6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]]
D :  [[ 1.62434536 -0.61175641]
 [-0.52817175 -1.07296862]]
E :  [[0.39676747 0.53881673]
 [0.41919451 0.6852195 ]]
F :  [[14 13  5  7]
 [10 10 12  4]]
```

```
+ Code + Text

# Reshaping 3X4 array to 2X2X3 array
A = np.array([[1, 2, 3, 4],
              [5, 6, 7, 8],
              [9, 1, 2, 3]])

new_A = A.reshape(2, 2, 3)

# Flatten array
B = np.array([[1, 2, 3], [4, 5, 6]])
flat_B= B.flatten()
# column_flat_B('C')

print ("\nOriginal array:\n", A)
print ("Reshaped array:\n", new_A)
print ("\nOriginal array:\n", B)
print ("Fattened array:\n", flat_B)
#print ("Column Fattened array:\n", column_flat_B)
```

```
Original array:
 [[1 2 3 4]
 [5 6 7 8]
 [9 1 2 3]]
Reshaped array:
 [[[1 2 3]
  [4 5 6]]

 [[7 8 9]
  [1 2 3]]]

Original array:
```

0s completed at 9:29 PM

BROADCASTING

```
+ Code + Text

[63] class MobilePhone:
       """This is a sample class to illustrate how Python classes work"""
       def __init__(self, name, is_android = False, screen_size = 4.3):
         self.name = name
         self.is_android = is_android
         self.screen_size = screen_size
         self.rating = -1

       def has_rating(self):
         return self.rating > -1

[64] new_phone = MobilePhone('iPhone 5s')

[65] type(new_phone)

     __main__.MobilePhone

     print(new_phone.name, new_phone.is_android, new_phone.screen_size)

     iPhone 5s False 4.3

[67] new_phone.screen_size = 4

[68] new_phone.has_rating()

     False

[69] new_phone.rating = 3.9
```

```
[71] class iPhone(MobilePhone):
       def __init__(self, name):
         MobilePhone.__init__(self, name, False, 4)

       def __str__(self):
         return self.name + " " + str(self.is_android) + " " + str(self.screen_size)
```

```
[72] new_iphone = iPhone('iPhone 5s')
```

```
[73] new_iphone.is_android
```

```
False
```

```
print(new_iphone)
```

```
iPhone 5s False 4
```

+ Code + Text

```
a = np.array([[1,1,1],[1,1,1], [1,1,1]])
b = np.array([2,2,2])
c = a + b
c1= a*b #element wise multiplication. This will also be broadcasted accordingly.

print(c)
print(c1)
```

```
[[3 3 3]
 [3 3 3]
 [3 3 3]]
[[2 2 2]
 [2 2 2]
 [2 2 2]]
```

1.15 Files I/O - The file object provides a set of methods to read and write on files. Some examples are given below.

1.15 Files I/O

OUTPUT

```
[76] # File open, read and write.

     file1 = open("sample.txt", "w+")
     file1.write("Welcome to the couse on Machine Learning at NIT Delhi.");
     file1.close()

     # Open a file
     file_read = open("sample.txt", "r")
     str = file_read.read(80);
     print("Name of the file: ", file_read.name)
     print("Read String from file is : ", str)
     # Close opend files
     file_read.close()
```

```
Name of the file:  sample.txt
Read String from file is :  Welcome to the couse on Machine Learning at NIT Delhi.
```

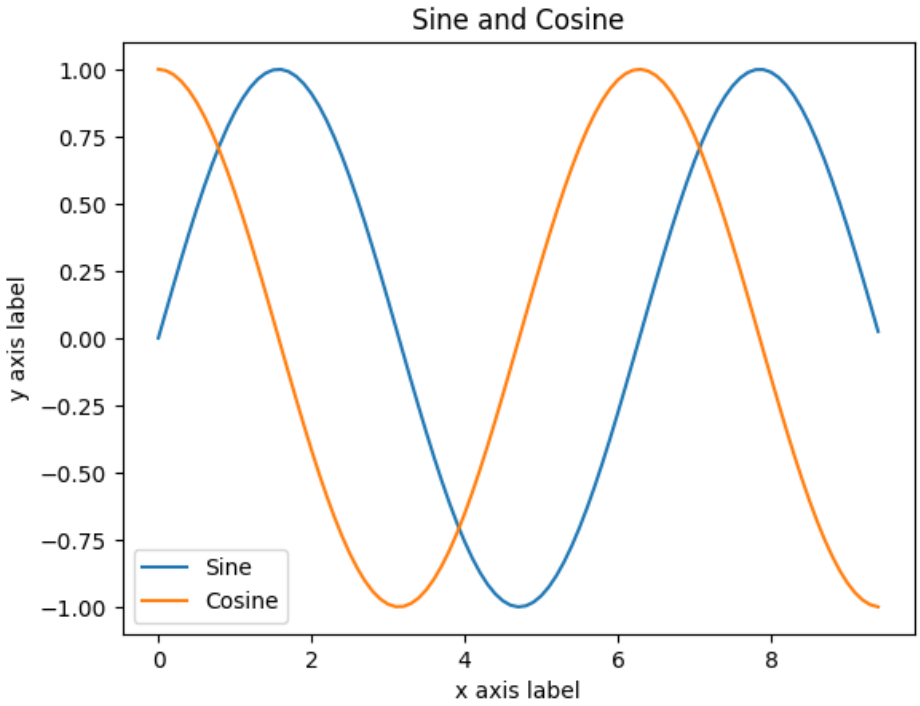
OBSERVATION

Just like in c++/c, file handling uses open(),read(),close() to open, read and close the file respectively.

1.16 Using matplotlib

OUTPUT

```
import numpy as np
import matplotlib.pyplot as plt

# Computes x and y coordinates for
# points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])

plt.show()
```



1.17 Using Images –

OUTPUT

```
import matplotlib.pyplot as plt
import cv2
import matplotlib.image as mpimg
img=mpimg.imread('NITD.jpeg')

print('Original Dimensions : ',img.shape)

scale_percent = 80 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

print('Resized Dimensions : ',resized.shape)
plt.figure(figsize=(4,10))
imgplot = plt.imshow(resized)
plt.show()
```

```
Original Dimensions :  (129, 129, 3)
Resized Dimensions :  (103, 103, 3)
```