Lab_1_1_basics_of_python

Aim : To understand basics of Python .

Completed By

Student Name :

Roll Number :

Branch : Electronics and Communication Engineering

Semester : 4

Lab: Signals and Systems (BEC 451)

Date of Completion ....

Importing libraries

In [1]:
```
# Simple imports
import math
import random
```

In [2]:
```
# importing specific functions from modules
# imports just the factorial function from math
from math import factorial

# imports all the functions from math
from math import *
```

In [3]:
```
# Giving aliases
# The Module name is alaised
import math as m

# The function name is alaised
from math import factorial as fact
```

In [4]:
```
# Calling imported functions
# If you import the module you have to call the functions from the module
import math
print (math.factorial(12))

# If you import the functions you can call the function as if it is in your pro
from random import randrange as rg
print (rg(23, 1000))
```

```
479001600
506
```

In [5]:
```
# Comments
# This is a python tutorial and a single line comment
''' This is a multiline comment
    pretty awesome!!
    Let me introduce you to Signals and Systems!'''
```

Out[5]:
```
' This is a multiline comment\n    pretty awesome!!\n    Let me introduce you
to Signals and Systems!'
```

Variables in Python

In [6]:
```
# Variables
msg = "Python!"  # String
v2 = 'Python!'   # Also String works same
v1 = 2           # Numbers
v3 = 3.564       # Floats / Doubles
v4 = True        # Boolean (True / False)
```

In [7]:
```
# print()
# automatically adds a newline
print (msg)
print (v2)
print (v1)
print (v3)
print (v4)
print ("Hello Python!")
```

```
Python!
Python!
2
3.564
True
Hello Python!
```

In [8]:
```
# Note: Both " and ' can be used to make strings. And this flexibility allows f

msg2 = 'Dr. Atul said, "I love Python!"'
msg3 = "After that Atul's Python Interpreter said it back to him!"
msg4 = 'Of Course she used the command `print("I love Signals and Systems")`'

print (msg2)
print (msg3)
print (msg4)
```

```
Dr. Atul said, "I love Python!"
After that Atul's Python Interpreter said it back to him!
Of Course she used the command `print("I love Signals and Systems")`
```

In [9]:
```
# input()
msg = input()
```

```
a
```

In [10]:
```
# input() with message
msg = input ("Provide some input: ")
print (msg)
```

```
Provide some input: 2
2
```

In [11]:
```
# Python takes every input as a string
# So, if required you can convert to the required type
msg = input("Enter a number: ")
print (type(msg))

msg = int(input ("Enter a number again, if not a number this will throw an erro
print (type(msg))
```

```
Enter a number: 2
<class 'str'>
Enter a number again, if not a number this will throw an error: 2
<class 'int'>
```

Basic Operators

In [12]:
```
# Basic Arithmetic operations
# Add
print (3 + 2)
print (3.4565 + 56.232)
print ('------------')

# Subtract
print (3 - 4)
print (34.56 - 3.78)
print ('------------')

# Multiply
print (4 * 3)
print (7.56 * 34)
print ('------------')

# Division
print (5 / 2)
print (5.0 / 2)
print (5 / 2.0)
print (25.0 / 5)
print ('------------')

# Exponents
print (4 ** 4)
print (5.67 ** 3)
print ('------------')

# Modulo
print (10%3)
print (10%11)
```

```
5
59.6885
------------
-1
30.78
------------
12
257.03999999999996
------------
2.5
2.5
2.5
5.0
------------
256
182.28426299999998
------------
1
10
```

Practice 1.1 : Find average Marks

#Write a program to input marks of three tests of a student (all integers). Then calculate and print the average of a

Practice 1.2 Find X raised to power N

Practice 1.3 Calculate area of a rectangle

t(res)ll

In [13]:
```
#prog prac 1.1
# Read input as sepcified in the question
# Print output as specified in the question
test1=int(input())
test2=int(input())
test3=int(input())
average=(test1+test2+test3)/3
print(average)
```

```
2
2
2
2.0
```

In [14]:
```
# prog prac 1.2
x=int(input())
n=int(input())
res=pow(x,n)
print(res)
```

```
3
3
27.0
```

In [15]:
```
# prog prac 1.3
# Left for your practice
```

Conditional Statements

In [16]:
```
# Check for specific input without storing it
if input("Enter something: ") == "something":
    print ("Something something")
else: print ("Not Something")
```

```
Enter something: 3
Not Something
```

Practice 1.4 Given an integer n, find if n is positive, negative or 0.

If n is positive, print "Positive" If n is negative, print "Negative" And if n is equal to 0, print "Zero".

In [17]:
```
# prog prac 1.4
n=int(input())
if n>0:
    print("Positive")
elif n<0:
    print("Negetive")
```

```
3
Positive
```

In [18]:
```
# if..else
v1 = 5
if v1 == 5:
    print (v1)
else:
    print ("v1 is not 5")
```

```
5
```

In [19]:
```
# if..elif..else
s1 = "Jennifer"
s2 = "loves"
s3 = "Python"
if s1 == "Python":
    print ("s1 is Python")
elif s2 == "Jennifer":
    print ("s2 is Atul")
elif s1 == "loves":
    print ("s1 is loves")
else:
    print ("Atul loves Python!")
```

```
Atul loves Python!
```

In [20]:
```
# One liner
v1 = 5
x = 10 if v1 == 5 else 13
print (x)
```

```
10
```

In [21]:
```
# Let's see the conditionals available
v1 = "Jennifer"
v2 = "Python"
v3 = 45
v4 = 67
v5 = 45

# Test for equality
print (v1 == v2)

# Test for greater than and greater than equal
print (v4 > v3)
print (v5 >= v2)

# Test for lesser than and lesser than equal
print (v4 < v3)
print (v5 <= v2)

# Inequality
print (v1 != v2)
```

```
False
True

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[21], line 13
     11 # Test for greater than and greater than equal
     12 print (v4 > v3)
---> 13 print (v5 >= v2)
     15 # Test for lesser than and lesser than equal
     16 print (v4 < v3)

TypeError: '>=' not supported between instances of 'int' and 'str'
```

In [ ]:
```
# Note:
v1 = 45
v2 = "45"
print (v1 == v2) # False
print (str(v1) == v2) # True
```

In [ ]:
```
# Ignore case when comparing two strings
s1 = "Atul"
s2 = "atul"

print (s1 == s2) # False
print (s1.lower() == s2.lower()) # True
# OR
print (s1.upper() == s2.upper()) # True
```

In [ ]:
```python
# Checking multiple conditions 'and' and 'or'
v1 = "Jennifer"
v2 = "Python"

# 'and' -> evaluates true when both conditions are True
print (v1 == "Jennifer" and v2 == "Python")
# 'or' -> evaluates true when any one condition is True
print (v1 == "Python" or v2 == "Python")
```

Note: When making comparisons with string with '>' or '<' The strings are compared lexographically.

In [ ]:
```python
s1 = "Atul"
s2 = "Python"

print (s1 > s2) # True -> since 'Atul' comes lexographically before 'Python'
```

In [ ]:
```python
# Check whether a value is in a list -> 'in'
l1 = [23, 45, 67, "Atul", "Python", 'A']

print (23 in l1)
print ('A' in l1)
print ("Python" in l1)
print (32 in l1)
```

In [ ]:
```python
# Putting it together
l1 = [23, 1, 'A', "Atul", 9.34]

# This is True, so the other statements are not checked
if 23 in l1 and 'B' not in l1: # Note: use of 'not'
    print ("1")
elif 23 >= l1[0]: # True
    print ("2")
elif 2.45 < l1[-1]: # True
    print ("3")
```

In [ ]:
```python
# Checking if list is empty
l1 = []
l2 = ["Jennifer"]

if l1:
    print (1)
elif l2:
    print (2)
```

Loops

In [ ]:
```
# One Liner while
v1 = 0
while v1 <= 40: v1 += 1
print (v1)
```

In [ ]:
```
# Terminate loop on a certain user input
# Note: The loop will break only when the user inputs 100
v1 = 1
while v1 != 100:
    v1 = int(input("Enter new v1: "))
    print ("v1 modified to: " + str(v1))
```

In [ ]:
```
# 'continue' -> continues to next iteration, skips all statements after it for
# Note: When 'v1' < 100 the last print statement is skipped and the control mov
while 1:
    print ("Iteration begins")
    v1 = int(input())
    if v1 == 100:
        break;
    elif v1 < 100:
        print ("v1 less than 100")
        continue;
    print ("Iteration complete")
```

In [ ]:
```
# Removing all instances of a specific value in list
l1 = ['A', 'B', 'C', 'D', 'A', 'E', 'Q', 'A', 'Z', 'A', 'Q', 'D', 'A']
while 'A' in l1: l1.remove('A')
print (l1)
```

In [ ]:
```
# Calculator using python
# for exit input 6

n=int(input())
while n!=6:
    if n <= 5 and n >= 1:
        a=int(input())
        b=int(input())
    if n==1:
        print(a+b)
    if n==2:
        print(a-b)
    if n==3:
        print(a*2)
    if n==4:
        print(a//b)
    if n==5:
        print(a%b)
    elif n < 1 or n > 6:
        print("Invalid Operation")
    n=int(input())
```

In [ ]:
```
# For loop
# Print number pyramid
n = int(input())
for i in range(1,n+1):
    count = 1
    for j in range(1,i):
        print(" ",end="")
        count = count + 1
    num = i
    for j in range(count,n+1):
        print(num,end="")
        num = num + 1
    print()
# increasing
for i in range(n-1,0,-1):
    count = 1
    for j in range(1,i):
        print(" ",end="")
        count = count + 1
    num = i
    for j in range(count,n+1):
        print(num,end="")
        num = num + 1
    print()
```

Function

In Python a function is defined using the def keyword:

In [ ]:
```
def my_function():
  print("Hello from a function")
```

Calling a Function

To call a function, use the function name followed by parenthesis:

In [ ]:
```
def my_function():
  print("Hello from a function")

my_function()
```

Passing a List as an Argument ¶

ou can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

E.g. if you send a List as an argument, it will still be a List when it reaches the function:

In [ ]:
```
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

In [ ]: