Python Basics

1)Arithmetic Operators

```
a = 46 # Initializing the value of a
b = 4 # Initializing the value of b
print("For a =", a, "and b =", b,"\nCalculate the following:")
# printing different results
print('1. Addition of two numbers: a + b =', a + b)
print('2. Subtraction of two numbers: a - b =', a - b)
print('3. Multiplication of two numbers: a * b =', a * b)
print('4. Division of two numbers: a / b =', a / b)
print('5. Floor division of two numbers: a // b =',a // b)
print('6. Reminder of two numbers: a mod b =', a % b)
print('7. Exponent of two numbers: a ^ b =',a ** b)
```

```
For a = 46 and b = 4
Calculate the following:
1. Addition of two numbers: a + b = 50
2. Subtraction of two numbers: a - b = 42
3. Multiplication of two numbers: a * b = 184
4. Division of two numbers: a / b = 11.5
5. Floor division of two numbers: a // b = 11
6. Reminder of two numbers: a mod b = 2
7. Exponent of two numbers: a ^ b = 4477456
```

2)Comparison Operators

```
a = 46 # Initializing the value of a
b = 4 # Initializing the value of b

print("For a =", a, "and b =", b,"\nCheck the following:")

print('1. Two numbers are equal or not:', a == b)
print('2. Two numbers are not equal or not:', a != b)
print('3. a is less than or equal to b:', a <= b)
print('4. a is greater than or equal to b:', a >= b)
print('5. a is greater b:', a > b)
print('6. a is less than b:', a < b)
```

```
For a = 46 and b = 4
Check the following:
1. Two numbers are equal or not: False
2. Two numbers are not equal or not: True
3. a is less than or equal to b: False
4. a is greater than or equal to b: True
5. a is greater b: True
6. a is less than b: False
```

3)Assignment Operators

```python
a = 34 # Initialize the value of a
b = 6 # Initialize the value of b
# printing the different results
print('a += b:', a + b)
print('a -= b:', a - b)
print('a *= b:', a * b)
print('a /= b:', a / b)
print('a %= b:', a % b)
print('a **= b:', a ** b)
print('a //= b:', a // b)
```

```
a += b: 40
a -= b: 28
a *= b: 204
a /= b: 5.666666666666667
a %= b: 4
a **= b: 1544804416
a //= b: 5
```

4)Bitwise Operators

```python
a = 7 # initializing the value of a
b = 8 # initializing the value of b
# printing different results
print('a & b :', a & b)
print('a | b :', a | b)
print('a ^ b :', a ^ b)
print('~a :', ~a)
print('a << b :', a << b)
print('a >> b :', a >> b)
```

```
a & b : 0
a | b : 15
a ^ b : 15
~a : -8
a << b : 1792
a >> b : 0
```

5)Logical Operators

```python
a = 7 # initializing the value of a
# printing different results
print("For a = 7, checking whether the following conditions are True or False:")
print('\"a > 5 and a < 7\" =>', a > 5 and a < 7)
print('\"a > 5 or a < 7\" =>', a > 5 or a < 7)
print('\"not (a > 5 and a < 7)\" =>', not(a > 5 and a < 7))
```

input

```
For a = 7, checking whether the following conditions are True or False:
"a > 5 and a < 7" => False
"a > 5 or a < 7" => True
"not (a > 5 and a < 7)" => True
```

6)Membership Operators

```
# initializing a list
myList = [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]
x = 31
y = 28
print("Given List:", myList)
if (x not in myList):
    print("x =", x,"is NOT present in the given list.")
else:
    print("x =", x,"is present in the given list.")
if (y in myList):
    print("y =", y,"is present in the given list.")
else:
    print("y =", y,"is NOT present in the given list.")
```

input

```
Given List: [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]
x = 31 is NOT present in the given list.
y = 28 is present in the given list.
```

7)Identity Operators

```
a = ["Rose", "Lotus"]
b = ["Rose", "Lotus"]
c = a
print("a is c => ", a is c)
print("a is not c => ", a is not c)
print("a is b => ", a is b)
print("a is not b => ", a is not b)
print("a == b => ", a == b)
print("a != b => ", a != b)
```

```
a is c =>  True
a is not c =>  False
a is b =>  False
a is not b =>  True
a == b =>  True
a != b =>  False
```

Python Reverse of the string

1)Using for loop

```
def reverse_string(str):
    str1 = "" # Declaring empty string to store the reversed string
    for i in str:
        str1 = i + str1
    return str1 # It will return the reverse string to the caller function
str = "JavaTpoint" # Given String
print("The original string is: ",str)
print("The reverse string is",reverse_string(str)) # Function call
```

```
The original string is:  JavaTpoint
The reverse string is tniopTavaJ
```

2)Using while loop

```
str = "JavaTpoint" # string variable
print ("The original string is : ",str)
reverse_String = "" # Empty String
count = len(str) # Find length of a string and save in count variable
while count > 0:
    reverse_String += str[ count - 1 ] # save the value of str[count-1] in reverseString
    count = count - 1 # decrement index
print ("The reversed string using a while loop is : ",reverse_String)# reversed string
```

input

```
The original string is :  JavaTpoint
The reversed string using a while loop is :  tniopTavaJ
```

3) Using the slice operator

```
def reverse(str):
    str = str[::-1]
    return str

s = "JavaTpoint"
print ("The original string is : ",s)
print ("The reversed string using extended slice operator is : ",reverse(s))
```

```
The original string is :  JavaTpoint
The reversed string using extended slice operator is :  tniopTavaJ
```

4) Using reverse function with join

```
def reverse(str):
    string = "".join(reversed(str)) # reversed() function inside the join()
    return string

s = "JavaTpoint"

print ("The original string is : ",s)
print ("The reversed string using reversed() is : ",reverse(s) )
```

```
The original string is :  JavaTpoint
The reversed string using reversed() is :  tniopTavaJ
```

5)Using recursion()

```
def reverse(str):
    if len(str) == 0:
        return str
    else:
        return reverse(str[1:]) + str[0]

str = "Bhargava Ram"
print ("The original string is : ", str)
print ("The reversed string(using recursion) is : ", reverse(str))
```

```
The original string is :  Bhargava Ram
The reversed string(using recursion) is :  maR avagrahB
```

How to read CSV file in Python

```python
import csv
with open(r'./example.csv') as csv_file:
    csv_read = csv.reader(csv_file, delimiter=',')
    count_line = 0
    for row in csv_read:
        if count_line == 0:
            print(f'Column names are {", ".join(row)}')
            count_line += 1
        else:
            print(f'\t{row[0]} roll number is: {row[1]} and department is: {row[2]}.')
            count_line += 1
    print(f'Processed {count_line} lines.')
```

```
[Running] python -u "c:\Users\Administrator\Desktop\1.PY"
Column names are Name , roll_no, department
    Bhargav roll number is: 1 and department is: CSD.
    samim roll number is: 2 and department is: IS.
    sahil roll number is: 3 and department is: AIML.
    parakram roll number is: 4 and department is: CSE.
    prabhakar roll number is: 5 and department is: IS1.
Processed 6 lines.
```

if statement

```python
num = int(input("enter the number:"))
if num%2 == 0:
    print("The Given number is an even number")
```

```
enter the number:10
The Given number is an even number
```

```
a = int (input("Enter a: "));
b = int (input("Enter b: "));
c = int (input("Enter c: "));
if a>b and a>c:
    print ("From the above three numbers given a is largest");
if b>a and b>c:
    print ("From the above three numbers given b is largest");
if c>a and c>b:
    print ("From the above three numbers given c is largest");
```

```
Enter a: 10
Enter b: 20
Enter c: 30
From the above three numbers given c is largest
```

if-else statement

```
age = int (input("Enter your age: "))
if age>=18:
    print("You are eligible to vote !!");
else:
    print("Sorry! you have to wait !!");
```

```
Enter your age: 22
You are eligible to vote !!
```

```python
num = int(input("enter the number:"))
if num%2 == 0:
    print("The Given number is an even number")
else:
    print("The Given Number is an odd number")
```

```
enter the number:43
The Given Number is an odd number
```

elif statement

```python
number = int(input("Enter the number?"))
if number==10:
    print("The given number is equals to 10")
elif number==50:
    print("The given number is equal to 50");
elif number==100:
    print("The given number is equal to 100");
else:
    print("The given number is not equal to 10, 50 or 100");
```

```
Enter the number?27
The given number is not equal to 10, 50 or 100
```

```python
marks = int(input("Enter the marks? "))
if marks > 85 and marks <= 100:
    print("Congrats ! you scored grade A ...")
elif marks > 60 and marks <= 85:
    print("You scored grade B + ...")
elif marks > 40 and marks <= 60:
    print("You scored grade B ...")
elif (marks > 30 and marks <= 40):
    print("You scored grade C ...")
else:
    print("Sorry you are fail ?")
```

```
Enter the marks? 99
Congrats ! you scored grade A ...
```

for Loop

```python
numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]
square = 0
squares = []
for value in numbers:
    square = value ** 2
    squares.append(square)
print("The list of squares is", squares)
```

input

```
The list of squares is [16, 4, 36, 49, 9, 25, 64, 100, 36, 1, 81, 4]
```

Using else Statement with for Loop

```python
string = "Python Loop"
for s in string:
    if s == "o":
        print("If block")
    else:
        print(s)
```

```
P
y
t
h
If block
n

L
If block
If block
p
```

```python
tuple_ = (3, 4, 6, 8, 9, 2, 3, 8, 9, 7)
for value in tuple_:
    if value % 2 != 0:
        print(value)
        print("It is the odd numbers present in the tuple")
    else:
        print(value)
        print("It is the even numbers present in the tuple")
```

```
3
It is the odd numbers present in the tuple
4
It is the even numbers present in the tuple
6
It is the even numbers present in the tuple
8
It is the even numbers present in the tuple
9
It is the odd numbers present in the tuple
2
It is the even numbers present in the tuple
3
It is the odd numbers present in the tuple
8
It is the even numbers present in the tuple
9
It is the odd numbers present in the tuple
7
It is the odd numbers present in the tuple
```

The range() Function

```
print(range(15))
print(list(range(15)))
print(list(range(4, 9)))
print(list(range(5, 25, 4)))
```

```
range(0, 15)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[4, 5, 6, 7, 8]
[5, 9, 13, 17, 21]
```

```python
tuple = ("Python", "Loops", "Sequence", "Condition", "Range")
for i in range(len(tuple)):
    print(tuple[i].upper())
```

```
PYTHON
LOOPS
SEQUENCE
CONDITION
RANGE
```

While Loop

```python
counter = 0
while counter < 10:
    counter = counter + 3
    print("Python Loops")
```

```
Python Loops
Python Loops
Python Loops
Python Loops
```

Using else Statement with while Loops

```python
counter = 0
while (counter < 10):
    counter = counter + 3
    print("Python Loops") # Executed untile condition is met
else:
    print("Code block inside the else statement")
```

```
Python Loops
Python Loops
Python Loops
Python Loops
Code block inside the else statement
```

Single statement while Block

```python
for string in "Python Loops":
    if string == "o" or string == "p" or string == "t":
        continue
    print('Current Letter:', string)
```

```
Current Letter: P
Current Letter: y
Current Letter: h
Current Letter: n
Current Letter:
Current Letter: L
Current Letter: s
```

Break Statement

```python
for string in "Python Loops":
    if string == 'L':
        break
    print('Current Letter: ', string)
```

```
Current Letter:  P
Current Letter:  y
Current Letter:  t
Current Letter:  h
Current Letter:  o
Current Letter:  n
Current Letter:
```

Pass Statement

```python
for string in "Python Loops":
    pass
print( 'Last Letter:', string)
```

```
Last Letter: s
```

Python for loop

```python
numbers = [3, 5, 23, 6, 5, 1, 2, 9, 8]
sum_ = 0
for num in numbers:
    sum_ = sum_ + num ** 2
print("The sum of squares is: ", sum_)
```

```
The sum of squares is:  774
```

The range() Function

```python
my_list = [3, 5, 6, 8, 4]
for iter_var in range( len( my_list ) ):
    my_list.append(my_list[iter_var] + 2)
print( my_list )
```

```
[3, 5, 6, 8, 4, 5, 7, 8, 10, 6]
```

Iterating by Using Index of Sequence

```python
numbers = [3, 5, 23, 6, 5, 1, 2, 9, 8]
sum_ = 0
for num in range( len(numbers) ):
    sum_ = sum_ + numbers[num] ** 2
print("The sum of squares is: ", sum_)
```

```
The sum of squares is:  774
```

Using else Statement with for Loop

```
student_name_1 = 'Itika'
student_name_2 = 'Parker'
records = {'Itika': 90, 'Arshia': 92, 'Peter': 46}
def marks( student_name ):
    for a_student in records:
        if a_student == student_name:
            return records[ a_student ]
            break
        else:
            return f'There is no student of name {student_name} in the records'
print( f"Marks of {student_name_1} are: ", marks( student_name_1 ) )
print( f"Marks of {student_name_2} are: ", marks( student_name_2 ) )
```

input

```
Marks of Itika are:  90
Marks of Parker are:  There is no student of name Parker in the records
```

Nested Loops

```
import random
numbers = [ ]
for val in range(0, 11):
    numbers.append( random.randint( 0, 11 ) )
for num in range( 0, 11 ):
    for i in numbers:
        if num == i:
            print( num, end = " " )
```

```
0 1 1 2 3 3 3 4 6 7 10
```

Python While Loops

```
i=1
while i<=10:
    print(i, end=' ')
    i+=1
```

```
1 2 3 4 5 6 7 8 9 10
```

```python
i=1
while i<51:
    if i%5 == 0 or i%7==0 :
        print(i, end=' ')
    i+=1
```

```
5 7 10 14 15 20 21 25 28 30 35 40 42 45 49 50
```

```python
num = 15
summation = 0
c = 1
while c <= num:
    summation = c**2 + summation
    c = c + 1 # incrementing the counter
print("The sum of squares is", summation)
```

```
The sum of squares is 1240
```

Prime Numbers and Python While Loop

```python
num = [34, 12, 54, 23, 75, 34, 11]
def prime_number(number):
    condition = 0
    iteration = 2
    while iteration <= number / 2:
        if number % iteration == 0:
            condition = 1
            break
        iteration = iteration + 1
    if condition == 0:
        print(f"{number} is a PRIME number")
    else:
        print(f"{number} is not a PRIME number")
for i in num:
    prime_number(i)
```

```
34 is not a PRIME number
12 is not a PRIME number
54 is not a PRIME number
23 is a PRIME number
75 is not a PRIME number
34 is not a PRIME number
11 is a PRIME number
```

Armstrong and Python While Loop

```python
n = int(input())
n1=str(n)
l=len(n1)
temp=n
s=0
while n!=0:
    r=n%10
    s=s+(r**l)
    n=n//10
if s==temp:
    print("It is an Armstrong number")
else:
    print("It is not an Armstrong number ")
```

```
153
It is an Armstrong number
```

Multiplication Table using While Loop

```python
num = 21
counter = 1
print("The Multiplication Table of: ", num)
while counter <= 10:
    ans = num * counter
    print (num, 'x', counter, '=', ans)
    counter += 1 # expression to increment the counter
```

```
The Multiplication Table of:  21
21 x 1 = 21
21 x 2 = 42
21 x 3 = 63
21 x 4 = 84
21 x 5 = 105
21 x 6 = 126
21 x 7 = 147
21 x 8 = 168
21 x 9 = 189
21 x 10 = 210
```

Python While Loop with List

```python
list_ = [3, 5, 1, 4, 6]
squares = []
while list_:
    squares.append( (list_.pop())**2)
print( squares )
```

```
[36, 16, 1, 25, 9]
```

```python
list_ = [3, 4, 8, 10, 34, 45, 67,80]
index = 0
while index < len(list_):
    element = list_[index]
    if element % 2 == 0:
        print('It is an even number')
    else:
        print('It is an odd number') # Print if the number is odd.
    index += 1
```

```
It is an odd number
It is an even number
It is an even number
It is an even number
It is an even number
It is an odd number
It is an odd number
It is an even number
```

```python
List_= ['Priya', 'Neha', 'Cow', 'To']
index = 0
while index < len(List_):
    element = List_[index]
    print(len(element))
    index += 1
```

```
5
4
3
2
```

Python While Loop Multiple Conditions

```python
num1 = 17
num2 = -12
while num1 > 5 and num2 < -5 :
    num1 -= 2
    num2 += 3
    print( (num1, num2) )
```

```
(15, -9)
(13, -6)
(11, -3)
```

```python
num1 = 17
num2 = -12
while num1 > 5 or num2 < -5 :
    num1 -= 2
    num2 += 3
    print( (num1, num2) )
```

```
(15, -9)
(13, -6)
(11, -3)
(9, 0)
(7, 3)
(5, 6)
```

```python
num1 = 9
num = 14
maximum_value = 4
counter = 0
while (counter < num1 or counter < num2) and not counter >= maximum_value:
    print(f"Number of iterations: {counter}")
    counter += 1
```

```
Number of iterations: 0
Number of iterations: 1
Number of iterations: 2
Number of iterations: 3
```

Loop Control Statements

```python
for string in "While Loops":
    if string == "o" or string == "i" or string == "e":
        continue
    print('Current Letter:', string)
```

```
Current Letter: W
Current Letter: h
Current Letter: l
Current Letter:
Current Letter: L
Current Letter: p
Current Letter: s
```

Break Statement

```python
for string in "Python Loops":
    if string == 'n':
        break
    print('Current Letter: ', string)
```

```
Current Letter:  P
Current Letter:  y
Current Letter:  t
Current Letter:  h
Current Letter:  o
```

Pass Statement

```python
for string in "Python Loops":
    pass
print( 'The Last Letter of given string is:', string)
```

```
The Last Letter of given string is: s
```

Python break statement

```python
my_list = [1, 2, 3, 4]
count = 1
for item in my_list:
    if item == 4:
        print("Item matched")
        count += 1
        break
print("Found at location", count)
```

input

```
Item matched
Found at location 2
```

```python
my_str = "python"
for char in my_str:
    if char == 'o':
        break
    print(char)
```

```
p
y
t
h
```

```python
i = 0;
while 1:
    print(i," ",end=""),
    i=i+1;
    if i == 10:
        break;
print("came out of while loop");
```

```
0  1  2  3  4  5  6  7  8  9  came out of while loop
```

```python
n = 2
while True:
    i = 1
    while i <= 10:
        print("%d X %d = %d\n" % (n, i, n * i))
        i += 1
    choice = int(input("Do you want to continue printing the table? Press 0 for no: "))
    if choice == 0:
        print("Exiting the program...")
        break
    n += 1
print("Program finished successfully.")
```

input

```
2 X 1 = 2

2 X 2 = 4

2 X 3 = 6

2 X 4 = 8

2 X 5 = 10

2 X 6 = 12

2 X 7 = 14

2 X 8 = 16

2 X 9 = 18

2 X 10 = 20

Do you want to continue printing the table? Press 0 for no: 0
Exiting the program...
Program finished successfully.
```

Python continue Statement

```python
for iterator in range(10, 21):
    if iterator == 15:
        continue
    print( iterator )
```

```
10
11
12
13
14
16
17
18
19
20
```

```python
string = "JavaTpoint"
iterator = 0
while iterator < len(string):
    if string[iterator] == 'a':
        iterator += 1
        continue
    print(string[ iterator ])
    iterator += 1
```

```
J
v
T
p
o
i
n
t
```

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sq_num = [num ** 2 for num in numbers if num % 2 == 0]
print(sq_num)
```

input

```
4, 16, 36, 64, 100]
```

Python String

```python
str1 = 'Hello Python'
print(str1)
str2 = "Hello Bhargav"
print(str2)

#Using triple quotes
str3 = '''Triple quotes are generally used for
    represent the multiline or
    docstring'''
print(str3)
```

```
Hello Python
Hello Bhargav
Triple quotes are generally used for
    represent the multiline or
    docstring
```

Strings indexing and splitting

```python
str = "HELLO"
print(str[0])
print(str[1])
print(str[2])
print(str[3])
print(str[4])
# It returns the IndexError because 6th index doesn't exist
print(str[6])
```

input

```
H
E
L
L
O
Traceback (most recent call last):
  File "/home/main.py", line 15, in <module>
    print(str[6])
          ~~~^^^
IndexError: string index out of range
```

```python
# Given String
str = "JAVATPOINT"
# Start 0th index to end
print(str[0:])
# Starts 1th index to 4th index
print(str[1:5])
# Starts 2nd index to 3rd index
print(str[2:4])
# Starts 0th to 2nd index
print(str[:3])
#Starts 4th to 6th index
print(str[4:7])
```

```
JAVATPOINT
AVAT
VA
JAV
TPO
```

```python
str = 'JAVATPOINT'
print(str[-1])
print(str[-3])
print(str[-2:])
print(str[-4:-1])
print(str[-7:-2])
# Reversing the given string
print(str[::-1])
print(str[-12])
```

```
T
I
NT
OIN
ATPOI
TNIOPTAVAJ
Traceback (most recent call last):
  File "/home/main.py", line 16, in <module>
    print(str[-12])
          ~~~^^^^^
IndexError: string index out of range
```

Reassigning Strings

```
8  str = "HELLO"
9  str[0] = "h"
10 print(str)
```

input

```
Traceback (most recent call last):
  File "/home/main.py", line 9, in <module>
    str[0] = "h"
    ~~~^^^
TypeError: 'str' object does not support item assignment
```

```
1 str = "HELLO"
2 print(str)
3 str = "hello"
4 print(str)
```

```
HELLO
hello
```

Deleting the String

```
1 str = "JAVATPOINT"
2 del str[1]
```

inpu

```
Traceback (most recent call last):
  File "/home/main.py", line 2, in <module>
    del str[1]
        ~~~^^^
TypeError: 'str' object doesn't support item deletion
```

```python
str1 = "JAVATPOINT"
del str1
print(str1)
```

input

```
Traceback (most recent call last):
  File "/home/main.py", line 3, in <module>
    print(str1)
          ^^^^
NameError: name 'str1' is not defined. Did you mean: 'str'?
```

String Operators

```python
str = "Hello"
str1 = " world"
print(str*3) # prints HelloHelloHello
print(str+str1)# prints Hello world
print(str[4]) # prints o
print(str[2:4]); # prints ll
print('w' in str) # prints false as w is not present in str
print('wo' not in str1) # prints false as wo is present in str1.
print(r'C://python37') # prints C://python37 as it is written
print("The string str : %s"%(str)) # prints The string str : Hello
```

input

```
HelloHelloHello
Hello world
o
ll
False
False
C://python37
The string str : Hello
```

Python String Formatting

```python
# using triple quotes
print('''They said, "What's there?"''')

# escaping single quotes
print('They said, "What\'s going on?"')

# escaping double quotes
print("They said, \"What's going on?\"")
```

input

```
They said, "What's there?"
They said, "What's going on?"
They said, "What's going on?"
```

```python
print("C:\\Users\\Bhargava Ram\\Python32\\Lib")
print("This is the \nmultiline quotes")
print("This is \x48\x45\x58 representation")
```

input

```
C:\Users\Bhargava Ram\Python32\Lib
This is the
multiline quotes
This is HEX representation
```

The format() method

```python
# Using Curly braces
print("{} and {} both are the best friend".format("Bhargav","Srikar"))

#Positional Argument
print("{1} and {0} best players ".format("Virat","Rohit"))

#Keyword Argument
print("{a},{b},{c}".format(a = "Samim", b = "Bhargav", c = "parakram"))
```

input

```
Bhargav and Srikar both are the best friend
Rohit and Virat best players
Samim,Bhargav,parakram
```

```python
Integer = 10
Float = 1.290
String = "Devansh"
print("Hi I am Integer ... My value is %d\nHi I am float ... My value is %f\nHi I am string ... My value is %s"%(Integer,Float,String))
```

input

```
Hi I am Integer ... My value is 10
Hi I am float ... My value is 1.290000
Hi I am string ... My value is Devansh
```

Python List

```python
list1 = [1, 2, "Python", "Program", 15.9]
list2 = ["Bhargav", "Srikar", "Samim", "Sahil"]

# printing the list
print(list1)
print(list2)

# printing the type of list
print(type(list1))
print(type(list2))
```

```
[1, 2, 'Python', 'Program', 15.9]
['Bhargav', 'Srikar', 'Samim', 'Sahil']
<class 'list'>
<class 'list'>
```

```python
a = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6 ]
b = [ 1, 2, 5, "Ram", 3.50, "Rahul", 6 ]
print(a == b)
```

```
False
```

```python
a = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6]
b = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6]
print(a == b)
```

```
True
```

```python
emp = [ "John", 102, "USA"]
Dep1 = [ "CS",10]
Dep2 = [ "IT",11]
HOD_CS = [ 10,"Mr. Holding"]
HOD_IT = [11, "Mr. Bewon"]
print("printing employee data ...")
print(" Name : %s, ID: %d, Country: %s" %(emp[0], emp[1], emp[2]))
print("printing departments ...")
print("Department 1:\nName: %s, ID: %d\n Department 2:\n Name: %s, ID: %s"%( Dep1[0], Dep2[1], Dep2[0], Dep2[1]))
print("HOD Details ....")
print("CS HOD Name: %s, Id: %d" %(HOD_CS[1], HOD_CS[0]))
print("IT HOD Name: %s, Id: %d" %(HOD_IT[1], HOD_IT[0]))
print(type(emp), type(Dep1), type(Dep2), type(HOD_CS), type(HOD_IT))
```

input

```
printing employee data ...
 Name : John, ID: 102, Country: USA
printing departments ...
Department 1:
Name: CS, ID: 11
 Department 2:
 Name: IT, ID: 11
HOD Details ....
CS HOD Name: Mr. Holding, Id: 10
IT HOD Name: Mr. Bewon, Id: 11
<class 'list'> <class 'list'> <class 'list'> <class 'list'> <class 'list'>
```

List Indexing and Splitting

```python
list = [1,2,3,4,5,6,7]
print(list[0])
print(list[1])
print(list[2])
print(list[3])
print(list[0:6])
print(list[:])
print(list[2:5])
print(list[1:6:2])
```

```
1
2
3
4
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6, 7]
[3, 4, 5]
[2, 4, 6]
```

```python
list = [1,2,3,4,5]
print(list[-1])
print(list[-3:])
print(list[:-1])
print(list[-3:-1])
```

```
5
[3, 4, 5]
[1, 2, 3, 4]
[3, 4]
```

Updating List Values

```python
list = [1, 2, 3, 4, 5, 6]
print(list)
# It will assign value to the value to the second index
list[2] = 10
print(list)
# Adding multiple-element
list[1:3] = [89, 78]
print(list)
# It will add value at the end of the list
list[-1] = 25
print(list)
```

```
[1, 2, 3, 4, 5, 6]
[1, 2, 10, 4, 5, 6]
[1, 89, 78, 4, 5, 6]
[1, 89, 78, 4, 5, 25]
```

Python List Operations

Repetition

```python
list1 = [12, 14, 16, 18, 20]
# repetition operator *
l = list1 * 2
print(l)
```

```
[12, 14, 16, 18, 20, 12, 14, 16, 18, 20]
```

Concatenation

```python
list1 = [12, 14, 16, 18, 20]
list2 = [9, 10, 32, 54, 86]
# concatenation operator +
l = list1 + list2
print(l)
```

```
[12, 14, 16, 18, 20, 9, 10, 32, 54, 86]
```

Length

```python
list1 = [12, 14, 16, 18, 20, 23, 27, 39, 40]
# finding length of the list
print(len(list1))
```

```
9
```

Iteration

```python
list1 = [12, 14, 16, 39, 40]
# iterating
for i in list1:
    print(i)
```

```
12
14
16
39
40
```

Membership

```python
list1 = [100, 200, 300, 400, 500]
print(600 in list1)
print(700 in list1)
print(1040 in list1)
print(300 in list1)
print(100 in list1)
print(500 in list1)
```

```
False
False
False
True
True
True
```

Iterating a List

```
list = ["John", "David", "James", "Jonathan"]
for i in list:
    print(i)
```

```
John
David
James
Jonathan
```

Adding Elements to the List

```
l =[]
n = int(input("Enter the number of elements in the list:"))
for i in range(0,n):
    l.append(input("Enter the item:"))
print("printing the list items..")
for i in l:
    print(i, end = "  ")
```

input

```
Enter the number of elements in the list:5
Enter the item:13
Enter the item:15
Enter the item:14
Enter the item:1
Enter the item:2
printing the list items..
13  15  14  1  2
```

Removing Elements from the List

```
list = [0,1,2,3,4]
print("printing original list: ");
for i in list:
    print(i,end=" ")
list.remove(2)
print("\nprinting the list after the removal of first element...")
for i in list:
    print(i,end=" ")
```

input

```
printing original list:
0 1 2 3 4
printing the list after the removal of first element...
0 1 3 4
```

len( )

```
list1 = [12, 16, 18, 20, 39, 40]
# finding length of the list
print(len(list1))
```

```
6
```

Max( )

```python
list1 = [103, 675, 321, 782, 200]
# large element in the list
print(max(list1))
```

```
782
```

Min( )

```python
list1 = [103, 675, 321, 782, 200]
# smallest element in the list
print(min(list1))
```

```
103
```

Python Tuples

```python
emptyTuple = ()
print("Empty tuple: ", emptyTuple)

# Creating a tuple having integers
integerTuple = (3, 6, 7, 10, 16, 23)
print("Tuple with integers: ", integerTuple)

# Creating a tuple having objects of different data types
mixedTuple = (6, "Javatpoint", 14.3)
print("Tuple with different data types: ", mixedTuple)

# Creating a nested tuple
nestedTuple = ("Javatpoint", {4: 5, 6: 2, 8:2}, (5, 3, 5, 6))
print("A nested tuple: ", nestedTuple)
```

input

```
Empty tuple:  ()
Tuple with integers:  (3, 6, 7, 10, 16, 23)
Tuple with different data types:  (6, 'Javatpoint', 14.3)
A nested tuple:  ('Javatpoint', {4: 5, 6: 2, 8: 2}, (5, 3, 5, 6))
```

```python
tupleWithoutParentheses = 6, 8.7, "Javatpoint", ["Python", "Tutorials"]
# Displaying the tuple created
print(tupleWithoutParentheses)
# Checking the data type of object tupleWithoutParentheses
print(type(tupleWithoutParentheses) )
# Trying to modify tupleWithoutParentheses
try:
    tupleWithoutParentheses[1] = 9.5
except:
    print("TypeError: Tuples are immutable data types and cannot be modified.")
```

input

```
(6, 8.7, 'Javatpoint', ['Python', 'Tutorials'])
<class 'tuple'>
TypeError: Tuples are immutable data types and cannot be modified.
```

```python
singleTuple = ("Arnold")
print("Type of the variable 'singleTuple' =>", type(singleTuple))  # returns <class 'str'>

# Creating a tuple that has only one element
singleTuple = ("Arnold",)
print("Type of the variable 'singleTuple' =>", type(singleTuple))  # returns <class 'tuple'>

# Creating a tuple without parentheses
singleTuple = "Arnold",
print("Type of the variable 'singleTuple' =>", type(singleTuple))  # returns <class 'tuple'>
```

input

```
Type of the variable 'singleTuple' => <class 'str'>
Type of the variable 'singleTuple' => <class 'tuple'>
Type of the variable 'singleTuple' => <class 'tuple'>
```

```python
sampleTuple = ("Apple", "Mango", "Banana", "Orange", "Guava", "Berries")

# accessing the elements of a tuple using indexing
print("First Element of the Given Tuple:", sampleTuple[0])
print("Second Element of the Given Tuple:", sampleTuple[1])
print("Third Element of the Given Tuple:", sampleTuple[2])
print("Forth Element of the Given Tuple:", sampleTuple[3])
print("Fifth Element of the Given Tuple:", sampleTuple[4])
print("Sixth Element of the Given Tuple:", sampleTuple[5])

# trying to access element index more than the length of a tuple
try:
    print("Seventh Element of the Given Tuple:", sampleTuple[6])
except Exception as e:
    print(e)

# trying to access elements through the index of floating data type
try:
    print("Accessing Second Element of the Given Tuple using floating-point index value:", sampleTuple[1.0])
except Exception as e:
    print(e)

# Creating a nested tuple
nestedTuple = ("Fruits", [4, 6, 2, 6], (6, 2, 6, 7))

# Accessing the index of a nested tuple
print(nestedTuple[0][3])
print(nestedTuple[1][1])
```

input

```
First Element of the Given Tuple: Apple
Second Element of the Given Tuple: Mango
Third Element of the Given Tuple: Banana
Forth Element of the Given Tuple: Orange
Fifth Element of the Given Tuple: Guava
Sixth Element of the Given Tuple: Berries
tuple index out of range
tuple indices must be integers or slices, not float
i
6
```

Negative Indexing

```python
sampleTuple = ("Apple", "Mango", "Banana", "Orange", "Guava", "Berries")

# Printing elements using negative indices
print("Element at -1 index: ", sampleTuple[-1])
print("Element at -2 index: ", sampleTuple[-2])
print("Element at -3 index: ", sampleTuple[-3])
print("Element at -4 index: ", sampleTuple[-4])
print("Element at -5 index: ", sampleTuple[-5])
print("Element at -6 index: ", sampleTuple[-6])

# Printing the range of elements using negative indices
print("Elements between -6 and -1 are: ", sampleTuple[-6:-1])
```

input

```
lement at -1 index:  Berries
lement at -2 index:  Guava
lement at -3 index:  Orange
lement at -4 index:  Banana
lement at -5 index:  Mango
lement at -6 index:  Apple
lements between -6 and -1 are:  ('Apple', 'Mango', 'Banana', 'Orange', 'Guava')
```

Slicing in Tuple

```python
sampleTuple = ("Apple", "Mango", "Banana", "Orange", "Guava", "Berries")

# Using slicing to access elements of the tuple
print("Elements between indices 1 and 5: ", sampleTuple[1:5])

# Using negative indexing in slicing
print("Elements between indices 0 and -3: ", sampleTuple[:-3])

# Printing the entire tuple by using the default start and end values
print("Entire tuple: ", sampleTuple[:])
```

input

```
Elements between indices 1 and 5:  ('Mango', 'Banana', 'Orange', 'Guava')
Elements between indices 0 and -3:  ('Apple', 'Mango', 'Banana')
Entire tuple:  ('Apple', 'Mango', 'Banana', 'Orange', 'Guava', 'Berries')
```

Deleting a Tuple

```python
sampleTuple = ("Apple", "Mango", "Banana", "Orange", "Guava", "Berries")
# printing the entire tuple for reference
print("Given Tuple:", sampleTuple)

# Deleting a particular element of the tuple using the del keyword
try:
    del sampleTuple[3]
    print(sampleTuple)
except Exception as e:
    print(e)

# Deleting the variable from the global space of the program using the del keyword
del sampleTuple

# Trying accessing the tuple after deleting it
try:
    print(sampleTuple)
except Exception as e:
    print(e)
```

input

```
Given Tuple: ('Apple', 'Mango', 'Banana', 'Orange', 'Guava', 'Berries')
'tuple' object doesn't support item deletion
name 'sampleTuple' is not defined
```

Changing the Elements in Tuple

```python
fruits_tuple = ("mango", "orange", "banana", "apple", "papaya")

# printing the tuple before update
print("Before Changing the Element in Tuple...")
print("Tuple =", fruits_tuple)

# converting the tuple into the list
fruits_list = list(fruits_tuple)

# changing the element of the list
fruits_list[2] = "grapes"
print("Converting", fruits_tuple[2], "=>", fruits_list[2])

# converting the list back into the tuple
fruits_tuple = tuple(fruits_list)

# printing the tuple after update
print("After Changing the Element in Tuple...")
print("Tuple =", fruits_tuple)
```

```
Before Changing the Element in Tuple...
Tuple = ('mango', 'orange', 'banana', 'apple', 'papaya')
Converting banana => grapes
After Changing the Element in Tuple...
Tuple = ('mango', 'orange', 'grapes', 'apple', 'papaya')
```

The len() Method

```python
sampleTuple = (5, 3, 6, 1, 2, 8, 7, 9, 0, 4)

# printing the tuple for reference
print("Given Tuple =>", sampleTuple)

# using the len() method to find the length of the tuple
length_of_tuple = len(sampleTuple)

# printing the result for the users
print("Number of Elements in the Given Tuple =>", length_of_tuple)
```

```
Given Tuple => (5, 3, 6, 1, 2, 8, 7, 9, 0, 4)
Number of Elements in the Given Tuple => 10
```

Python Functions

Illustration of a User-Defined Function

```python
def square( num ):
    """
    This function computes the square of the number.
    """
    return num**2
object_ = square(6)
print( "The square of the given number is: ", object_ )
```

```
The square of the given number is:  36
```

Calling a Function

```python
def a_function( string ):
    "This prints the value of length of string"
    return len(string)

# Calling the function we defined
print( "Length of the string Functions is: ", a_function( "Functions" ) )
print( "Length of the string Python is: ", a_function( "Python" ) )
```

```
Length of the string Functions is:  9
Length of the string Python is:  6
```

Pass by Reference vs. Pass by Value

```python
def square( item_list ):
    '''''''This function will find the square of items in the list'''
    squares = [ ]
    for l in item_list:
        squares.append( l**2 )
    return squares

# calling the defined function
my_list = [17, 52, 8];
my_result = square( my_list )
print( "Squares of the list are: ", my_result )
```

```
Squares of the list are:  [289, 2704, 64]
```

Function Arguments

1) Default Arguments

```python
def function( n1, n2 = 20 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)


# Calling the function and passing only one argument
print( "Passing only one argument" )
function(30)

# Now giving two arguments to the function
print( "Passing two arguments" )
function(50,30)
```

```
Passing only one argument
number 1 is:  30
number 2 is:  20
Passing two arguments
number 1 is:  50
number 2 is:  30
```

2) Keyword Arguments

```python
def function( n1, n2 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)

# Calling function and passing arguments without using keyword
print( "Without using keyword" )
function( 50, 30)

# Calling function and passing arguments using keyword
print( "With using keyword" )
function( n2 = 50, n1 = 30)
```

```
Without using keyword
number 1 is:  50
number 2 is:  30
With using keyword
number 1 is:  30
number 2 is:  50
```

3) Required Arguments

```python
def function( n1, n2 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)

# Calling function and passing two arguments out of order, we need num1 to be 20 and num2 to be 30
print( "Passing out of order arguments" )
function( 30, 20 )

# Calling function and passing only one argument
print( "Passing only one argument" )
try:
    function( 30 )
except:
    print( "Function needs two positional arguments" )
```

```
Passing out of order arguments
number 1 is:  30
number 2 is:  20
Passing only one argument
Function needs two positional arguments
```

4) Variable-Length Arguments

```python
def function( *args_list ):
    ans = []
    for l in args_list:
        ans.append( l.upper() )
    return ans
# Passing args arguments
object = function('Python', 'Functions', 'tutorial')
print( object )

# defining a function
def function( **kargs_list ):
    ans = []
    for key, value in kargs_list.items():
        ans.append([key, value])
    return ans
# Paasing kwargs arguments
object = function(First = "Python", Second = "Functions", Third = "Tutorial")
print(object)
```

```
['PYTHON', 'FUNCTIONS', 'TUTORIAL']
[['First', 'Python'], ['Second', 'Functions'], ['Third', 'Tutorial']]
```

return Statement

```python
def square( num ):
    return num**2

# Calling function and passing arguments.
print( "With return statement" )
print( square( 52 ) )

# Defining a function without return statement
def square( num ):
    num**2

# Calling function and passing arguments.
print( "Without return statement" )
print( square( 52 ) )
```

```
With return statement
2704
Without return statement
None
```

The Anonymous Functions

```python
lambda_ = lambda argument1, argument2: argument1 + argument2;

# Calling the function and passing values
print( "Value of the function is : ", lambda_( 20, 30 ) )
print( "Value of the function is : ", lambda_( 40, 50 ) )
```

```
Value of the function is :  50
Value of the function is :  90
```

Scope and Lifetime of Variables

```python
def number( ):
    num = 50
    print( "Value of num inside the function: ", num)

num = 10
number()
print( "Value of num outside the function:", num)
```

```
Value of num inside the function:  50
Value of num outside the function: 10
```

Python Capability inside Another Capability

```python
def word():
    string = 'Python functions tutorial'
    x = 5
    def number():
        print( string )
        print( x )

    number()
word()
```

```
Python functions tutorial
5
```

Python Built-in Functions

Python abs() Function

```python
integer = -20
print('Absolute value of -40 is:', abs(integer))

#  floating number
floating = -20.83
print('Absolute value of -40.83 is:', abs(floating))
```

```
Absolute value of -40 is: 20
Absolute value of -40.83 is: 20.83
```

Python all() Function

Python all() Function Example

```python
k = [1, 3, 4, 6]
print(all(k))

# all values false
k = [0, False]
print(all(k))

# one false value
k = [1, 3, 7, 0]
print(all(k))

# one true value
k = [0, False, 5]
print(all(k))

# empty iterable
k = []
print(all(k))
```

```
True
False
False
False
True
```

Python bin() Function

```
x =  10
y =  bin(x)
print (y)
```

```
0b1010
```

Python bool()

```
test1 = []
print(test1,'is',bool(test1))
test1 = [0]
print(test1,'is',bool(test1))
test1 = 0.0
print(test1,'is',bool(test1))
test1 = None
print(test1,'is',bool(test1))
test1 = True
print(test1,'is',bool(test1))
test1 = 'Easy string'
print(test1,'is',bool(test1))
```

```
[] is False
[0] is True
0.0 is False
None is False
True is True
Easy string is True
```

Python compile() Function

```
code_str = 'x=5\ny=10\nprint("sum =",x+y)'
code = compile(code_str, 'sum.py', 'exec')
print(type(code))
exec(code)
exec(x)
```

```
<class 'code'>
sum = 15
```

Python exec() Function

```
x = 8
exec('print(x==8)')
exec('print(x+4)')
```

```
True
12
```

Python any() Function

```python
l = [4, 3, 2, 0]
print(any(l))

l = [0, False]
print(any(l))

l = [0, False, 5]
print(any(l))

l = []
print(any(l))
```

```
True
False
True
False
```

Python float()

```python
# for integers
print(float(9))

# for floats
print(float(8.19))

# for string floats
print(float("-24.27"))

# for string floats with whitespaces
print(float("     -17.19\n"))

# string float error
print(float("xyz"))
```

```
9.0
8.19
-24.27
-17.19
Traceback (most recent call last):
  File "/home/main.py", line 14, in <module>
    print(float("xyz"))
          ^^^^^^^^^^^^
ValueError: could not convert string to float: 'xyz'
```

Python format() Function

```python
# d, f and b are a type

# integer
print(format(123, "d"))

# float arguments
print(format(123.4567898, "f"))

# binary format
print(format(12, "b"))
```

```
123
123.456790
1100
```

Python hasattr() Function

```python
l = [4, 3, 2, 0]
print(any(l))

l = [0, False]
print(any(l))

l = [0, False, 5]
print(any(l))

l = []
print(any(l))
```

```
True
False
True
False
```

Python iter() Function Example

```python
# list of numbers
list = [1,2,3,4,5]

listIter = iter(list)

# prints '1'
print(next(listIter))

# prints '2'
print(next(listIter))

# prints '3'
print(next(listIter))

# prints '4'
print(next(listIter))

# prints '5'
print(next(listIter))
```

```
1
2
3
4
5
```

Python len() Function

```python
strA = 'Python'
print(len(strA))
```

```
6
```

Python list()

```python
# empty list
print(list())

# string
String = 'abcde'
print(list(String))

# tuple
Tuple = (1,2,3,4,5)
print(list(Tuple))
# list
List = [1,2,3,4,5]
print(list(List))
```

```
[]
['a', 'b', 'c', 'd', 'e']
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```

Python memoryview() Function

```python
#A random bytearray
randomByteArray = bytearray('ABC', 'utf-8')

mv = memoryview(randomByteArray)

# access the memory view's zeroth index
print(mv[0])

# It create byte from memory view
print(bytes(mv[0:2]))

# It create list from memory view
print(list(mv[0:3]))
```

```
65
b'AB'
[65, 66, 67]
```

Python object()

```python
python = object()

print(type(python))
print(dir(python))
```

```
input
<class 'object'>
['__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subc
lass__', '__le__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__']
```

Python hash() Function

```python
# Calling function
result = hash(21) # integer value
result2 = hash(22.2) # decimal value
# Displaying result
print(result)
print(result2)
```

```
21
461168601842737174
```

Python pow() Function

```python
# positive x, positive y (x**y)
print(pow(4, 2))

# negative x, positive y
print(pow(-4, 2))

# positive x, negative y (x**-y)
print(pow(4, -2))

# negative x, negative y
print(pow(-4, -2))
```

```
16
16
0.0625
0.0625
```

Python reversed() Function

```python
# for string
String = 'Java'
print(list(reversed(String)))

# for tuple
Tuple = ('J', 'a', 'v', 'a')
print(list(reversed(Tuple)))

# for range
Range = range(8, 12)
print(list(reversed(Range)))

# for list
List = [1, 2, 7, 5]
print(list(reversed(List)))
```

```
['a', 'v', 'a', 'J']
['a', 'v', 'a', 'J']
[11, 10, 9, 8]
[5, 7, 2, 1]
```

Python issubclass() Function

```python
class Rectangle:
  def __init__(rectangleType):
    print('Rectangle is a ', rectangleType)

class Square(Rectangle):
  def __init__(self):
    Rectangle.__init__('square')

print(issubclass(Square, Rectangle))
print(issubclass(Square, list))
print(issubclass(Square, (list, Rectangle)))
print(issubclass(Rectangle, (list, Rectangle)))
```

```
True
False
True
True
```

Python tuple() Function

```python
t1 = tuple()
print('t1=', t1)

# creating a tuple from a list
t2 = tuple([1, 6, 9])
print('t2=', t2)

# creating a tuple from a string
t1 = tuple('Java')
print('t1=',t1)

# creating a tuple from a dictionary
t1 = tuple({4: 'four', 5: 'five'})
print('t1=',t1)
```

```
t1= ()
t2= (1, 6, 9)
t1= ('J', 'a', 'v', 'a')
t1= (4, 5)
```

Python zip() Function

```python
numList = [4,5, 6]
strList = ['four', 'five', 'six']

# No iterables are passed
result = zip()

# Converting itertor to list
resultList = list(result)
print(resultList)

# Two iterables are passed
result = zip(numList, strList)

# Converting itertor to set
resultSet = set(result)
print(resultSet)
```

```
[]
{(6, 'six'), (5, 'five'), (4, 'four')}
```

Python Lambda Functions

```python
add = lambda num: num + 4
print( add(6) )
```

```
10
```

```python
def add( num ):
    return num + 4
print( add(6) )
```

```
10
```

```python
a = lambda x, y : (x * y)
print(a(4, 5))
```

```
10
```

```python
a = lambda x, y, z : (x + y + z)
print(a(4, 5, 5))
```

```
14
```

What's the Distinction Between Lambda and Def Functions?

```python
def reciprocal( num ):
    return 1 / num

lambda_reciprocal = lambda num: 1 / num

# using the function defined by def keyword
print( "Def keyword: ", reciprocal(6) )

# using the function defined by lambda keyword
print( "Lambda keyword: ", lambda_reciprocal(6) )
```

```
14
```

Using Lambda Function with filter()

```python
list_ = [35, 12, 69, 55, 75, 14, 73]
odd_list = list(filter( lambda num: (num % 2 != 0) , list_ ))
print('The list of odd number is:',odd_list)
```

```
The list of odd number is: [35, 69, 55, 75, 73]
```

Using Lambda Function with map()

```python
numbers_list = [2, 4, 5, 1, 3, 7, 8, 9, 10]
squared_list = list(map( lambda num: num ** 2 , numbers_list ))
print( 'Square of each number in the given list:' ,squared_list )
```

```
Square of each number in the given list: [4, 16, 25, 1, 9, 49, 64, 81, 100]
```

Using Lambda Function with List Comprehension

```python
squares = [lambda num = num: num ** 2 for num in range(0, 11)]
for square in squares:
    print('The square value of all numbers from 0 to 10:',square(), end = " ")
```

```
The square value of all numbers from 0 to 10: 0 The square value of all numbers from 0 to 10: 1 The square value of all numbers from 0 to 10: 4 The square value of all
numbers from 0 to 10: 9 The square value of all numbers from 0 to 10: 16 The square value of all numbers from 0 to 10: 25 The square value of all numbers from 0 to 10
: 36 The square value of all numbers from 0 to 10: 49 The square value of all numbers from 0 to 10: 64 The square value of all numbers from 0 to 10: 81 The square valu
e of all numbers from 0 to 10: 100
```

Using Lambda Function with if-else

```python
Minimum = lambda x, y : x if (x < y) else y
print('The greater number is:', Minimum( 35, 74 ))
```

```
The greater number is: 35
```

Using Lambda with Multiple Statements

```python
my_List = [ [3, 5, 8, 6], [23, 54, 12, 87], [1, 2, 4, 12, 5] ]
# sorting every sublist of the above list
sort_List = lambda num : ( sorted(n) for n in num )
# Getting the third largest number of the sublist
third_Largest = lambda num, func : [ l[ len(l) - 2] for l in func(num)]
result = third_Largest( my_List, sort_List)
print('The third largest number from every sub list is:', result )
```

```
The third largest number from every sub list is: [6, 54, 5]
```

Python Modules

example_module.py main_program.py

python > main_program.py

```python
import example_module
result = example_module.square( 4 )
print("By using the module square of number is:",result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
[Running] python -u "c:\Users\Administrator\Desktop\python\main_program.py"
By using the module square of number is: 16

[Done] exited with code=0 in 0.716 seconds
```

Importing and also Renaming

```python
import math
print( "The value of euler's number is", math.e )
```

```
The value of euler's number is 2.718281828459045
```

Python from...import Statement

```python
from math import e, tau
print( "The value of tau constant is: ", tau )
print( "The value of the euler's number is: ", e )
```

```
The value of tau constant is:  6.283185307179586
The value of the euler's number is:  2.718281828459045
```

Import all Names - From import * Statement

```python
from math import *
# Here, we are accessing functions of math module without using the dot operator
print( "Calculating square root: ", sqrt(25) )
# here, we are getting the sqrt method and finding the square root of 25
print( "Calculating tangent of an angle: ", tan(pi/6) )
```

```
Calculating square root:  5.0
Calculating tangent of an angle:  0.5773502691896257
```

Locating Path of Modules

```python
import sys
# Here, we are printing the path using sys.path
print("Path of the sys module in the system is:", sys.path)
```

```
Path of the sys module in the system is: ['/home', '/usr/lib/python312.zip', '/usr/lib/python3.12', '/usr/lib/python3.12/lib-dynload', '/usr/local/lib/python3.12/dist-packages', '/usr/lib/python3/dist-packages']
```

The dir() Built-in Function

```python
print( "List of functions:\n ", dir( str ), end=", " )
```

```
List of functions:
 ['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill'],
```

Namespaces and Scoping

```python
Number = 204
def AddNumber():  # here, we are defining a function with the name Add Number
    # Here, we are accessing the global namespace
    global Number
    Number = Number + 200
print("The number is:", Number)
# here, we are printing the number after performing the addition
AddNumber()    # here, we are calling the function
print("The number is:", Number)
```

```
The number is: 204
The number is: 404
```