Python Basics

1)Arithmetic Operators

```python
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

```python
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

```python
# 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

```python
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

```python
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

```python
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

```python
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

```python
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()

```python
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
```
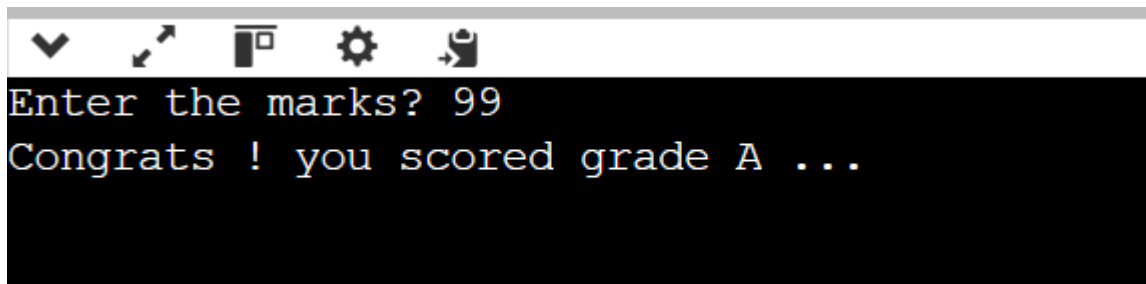
elif statement

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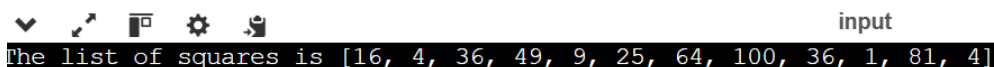
```python
marks = int(input("Enter the marks? "))
if marks > 85 and marks <= 100:
    print("Congrats ! you scored grade A ...")
elif marks > 60 and marks <= 85:
    print("You scored grade B + ...")
elif marks > 40 and marks <= 60:
    print("You scored grade B ...")
elif (marks > 30 and marks <= 40):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

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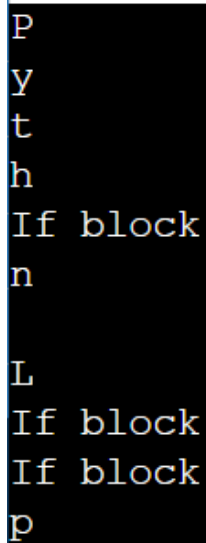
for Loop

```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

```python
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]
```

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

```
PYTHON
LOOPS
SEQUENCE
CONDITION
RANGE
```
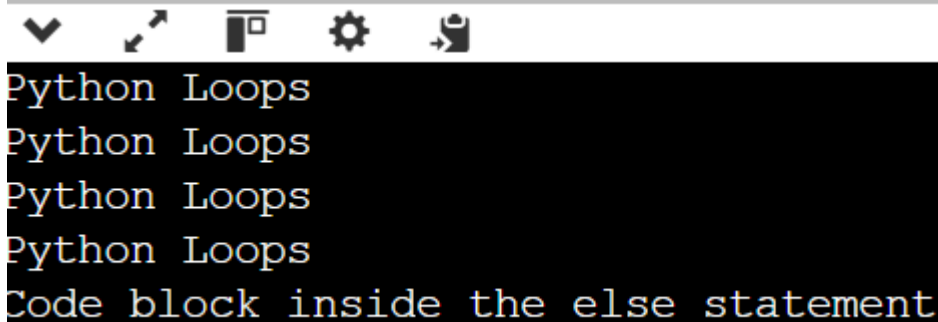
While Loop

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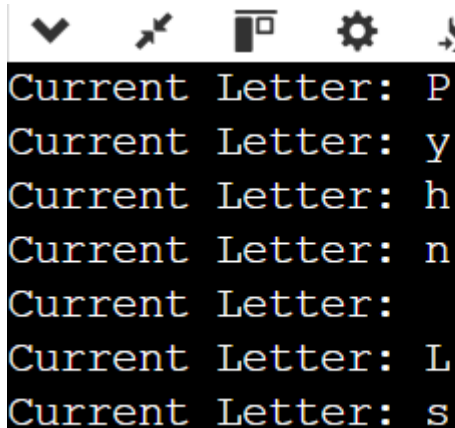
```
counter = 0
while (counter < 10):
    counter = counter + 3
    print("Python Loops") # Executed untile condition is met
else:
    print("Code block inside the else statement")
```

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

Single statement while Block

```
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

```
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

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

```
Last Letter: s
```

Python for loop

```
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

```
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

```
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

```python
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

```python
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

```python
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

```
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

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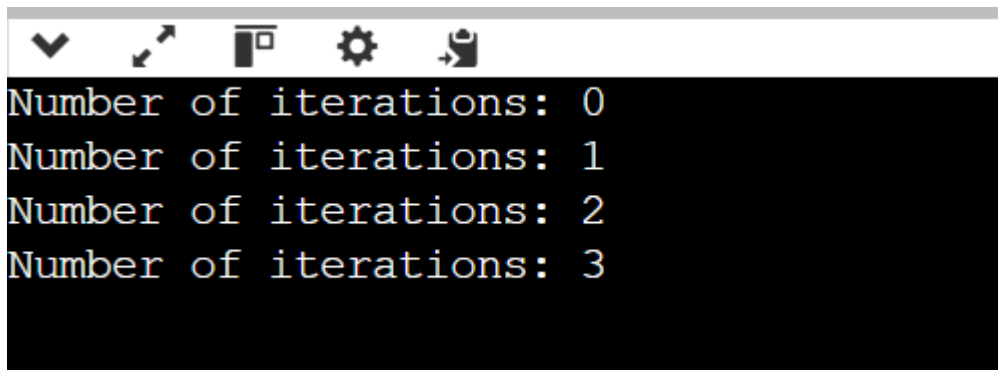
```python
num1 = 17
num2 = -12
while num1 > 5 or num2 < -5 :
    num1 -= 2
    num2 += 3
    print( (num1, num2) )
```

```
(15, -9)
(13, -6)
(11, -3)
(9, 0)
(7, 3)
(5, 6)
```
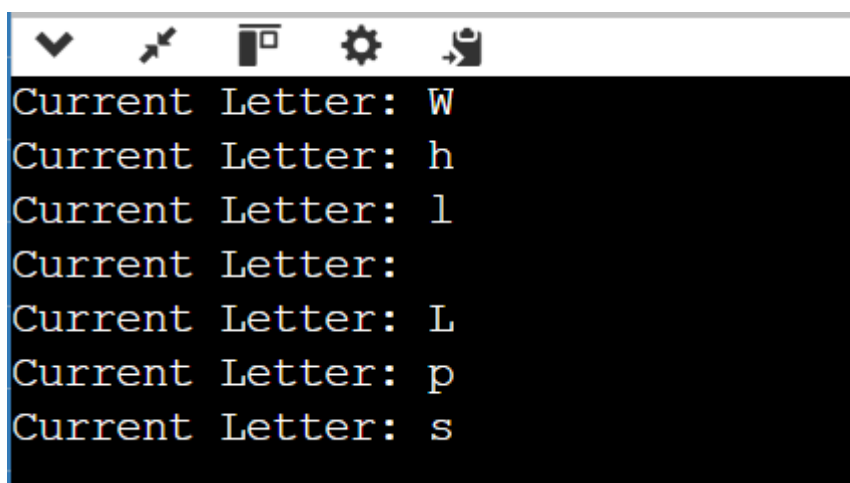
```python
num1 = 9
num = 14
maximum_value = 4
counter = 0
while (counter < num1 or counter < num2) and not counter >= maximum_value:
    print(f"Number of iterations: {counter}")
    counter += 1
```

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

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

Python break statement

```python
 8  my_list = [1, 2, 3, 4]
 9  count = 1
10 ▾ for item in my_list:
11 ▾     if item == 4:
12             print("Item matched")
13             count += 1
14             break
15  print("Found at location", count)
```

input

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

```python
 8   my_str = "python"
 9 ▾ for char in my_str:
10 ▾     if char == 'o':
11             break
12         print(char)
```

```
p
y
t
h
```

```python
 8   i = 0;
 9 ▾ while 1:
10       print(i," ",end=""),
11       i=i+1;
12 ▾     if i == 10:
13             break;
14   print("came out of while loop");
```

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

```
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

```
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
```

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

input

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

Python String

```
8  str1 = 'Hello Python'
9  print(str1)
10 str2 = "Hello Bhargav"
11 print(str2)
12
13 #Using triple quotes
14 str3 = '''Triple quotes are generally used for
15      represent the multiline or
16      docstring'''
17 print(str3)
```

in

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

Strings indexing and splitting

```
8  str = "HELLO"
9  print(str[0])
10 print(str[1])
11 print(str[2])
12 print(str[3])
13 print(str[4])
14 # It returns the IndexError because 6th index doesn't exist
15 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
7  """
8  # Given String
9  str = "JAVATPOINT"
10  # Start 0th index to end
11  print(str[0:])
12  # Starts 1th index to 4th index
13  print(str[1:5])
14  # Starts 2nd index to 3rd index
15  print(str[2:4])
16  # Starts 0th to 2nd index
17  print(str[:3])
18  #Starts 4th to 6th index
19  print(str[4:7])
```

```
JAVATPOINT
AVAT
VA
JAV
TPO
```

```python
7  """
8  str = 'JAVATPOINT'
9  print(str[-1])
10  print(str[-3])
11  print(str[-2:])
12  print(str[-4:-1])
13  print(str[-7:-2])
14  # Reversing the given string
15  print(str[::-1])
16  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
```

```
1   str1 = "JAVATPOINT"
2   del str1
3   print(str1)
```

```
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

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

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

Python String Formatting

```
1   # using triple quotes
2   print('''They said, "What's there?"''')
3
4   # escaping single quotes
5   print('They said, "What\'s going on?"')
6
7   # escaping double quotes
8   print("They said, \"What's going on?\"")
```

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

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

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

The format() method

```
1  # Using Curly braces
2  print("{} and {} both are the best friend".format("Bhargav","Srikar"))
3
4  #Positional Argument
5  print("{1} and {0} best players ".format("Virat","Rohit"))
6
7  #Keyword Argument
8  print("{a},{b},{c}".format(a = "Samim", b = "Bhargav", c = "parakram"))
```

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

```
1  Integer = 10
2  Float = 1.290
3  String = "Devansh"
4  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))
```

```
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

```
1   list1 = [1, 2, "Python", "Program", 15.9]
2   list2 = ["Bhargav", "Srikar", "Samim", "Sahil"]
3
4   # printing the List
5   print(list1)
6   print(list2)
7
8   # printing the type of List
9   print(type(list1))
10  print(type(list2))
```

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

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

False

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

True

```
1   emp = [ "John", 102, "USA"]
2   Dep1 = [ "CS",10]
3   Dep2 = [ "IT",11]
4   HOD_CS = [ 10,"Mr. Holding"]
5   HOD_IT = [11, "Mr. Bewon"]
6   print("printing employee data ...")
7   print(" Name : %s, ID: %d, Country: %s" %(emp[0], emp[1], emp[2]))
8   print("printing departments ...")
9   print("Department 1:\nName: %s, ID: %d\n Department 2:\n Name: %s, ID: %s"%( Dep1[0], Dep2[1], Dep2[0], Dep2[1]))
10  print("HOD Details ....")
11  print("CS HOD Name: %s, Id: %d" %(HOD_CS[1], HOD_CS[0]))
12  print("IT HOD Name: %s, Id: %d" %(HOD_IT[1], HOD_IT[0]))
13  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

```
1  list = [1,2,3,4,5,6,7]
2  print(list[0])
3  print(list[1])
4  print(list[2])
5  print(list[3])
6  print(list[0:6])
7  print(list[:])
8  print(list[2:5])
9  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]
```

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

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

Updating List Values

```
1   list = [1, 2, 3, 4, 5, 6]
2   print(list)
3   # It will assign value to the value to the second index
4   list[2] = 10
5   print(list)
6   # Adding multiple-element
7   list[1:3] = [89, 78]
8   print(list)
9   # It will add value at the end of the list
10  list[-1] = 25
11  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

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

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

Concatenation

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

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

Length

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

```
9
```

Iteration

```
1  list1 = [12, 14, 16, 39, 40]
2  # iterating
3  for i in list1:
4      print(i)
5
```

```
12
14
16
39
40
```

Membership

```
1  list1 = [100, 200, 300, 400, 500]
2  print(600 in list1)
3  print(700 in list1)
4  print(1040 in list1)
5  print(300 in list1)
6  print(100 in list1)
7  print(500 in list1)
```

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

Iterating a List

```
2  list = ["John", "David", "James", "Jonathan"]
3  for i in list:
4      print(i)
5
```

```
John
David
James
Jonathan
```

Adding Elements to the List

```
1  l =[]
2  n = int(input("Enter the number of elements in the list:"))
3  for i in range(0,n):
4      l.append(input("Enter the item:"))
5  print("printing the list items..")
6  for i in l:
7      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

```
1  list = [0,1,2,3,4]
2  print("printing original list: ");
3  for i in list:
4      print(i,end=" ")
5  list.remove(2)
6  print("\nprinting the list after the removal of first element...")
7  for i in list:
8      print(i,end=" ")
9
```

input

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

len( )

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

```
6
```

Max( )

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

```
782
```

Min( )

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

```
103
```

Python Tuples

```
1   emptyTuple = ()
2   print("Empty tuple: ", emptyTuple)
3
4   # Creating a tuple having integers
5   integerTuple = (3, 6, 7, 10, 16, 23)
6   print("Tuple with integers: ", integerTuple)
7
8   # Creating a tuple having objects of different data types
9   mixedTuple = (6, "Javatpoint", 14.3)
10  print("Tuple with different data types: ", mixedTuple)
11
12  # Creating a nested tuple
13  nestedTuple = ("Javatpoint", {4: 5, 6: 2, 8:2}, (5, 3, 5, 6))
14  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))
```

```
1  tupleWithoutParentheses = 6, 8.7, "Javatpoint", ["Python", "Tutorials"]
2  # Displaying the tuple created
3  print(tupleWithoutParentheses)
4  # Checking the data type of object tupleWithoutParentheses
5  print(type(tupleWithoutParentheses) )
6  # Trying to modify tupleWithoutParentheses
7  try:
8      tupleWithoutParentheses[1] = 9.5
9  except:
10     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.
```

```
1  singleTuple = ("Arnold")
2  print("Type of the variable 'singleTuple' =>", type(singleTuple))  # returns <class 'str'>
3
4  # Creating a tuple that has only one element
5  singleTuple = ("Arnold",)
6  print("Type of the variable 'singleTuple' =>", type(singleTuple))  # returns <class 'tuple'>
7
8  # Creating a tuple without parentheses
9  singleTuple = "Arnold",
10 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'>
```

```
1  sampleTuple = ("Apple", "Mango", "Banana", "Orange", "Guava", "Berries")
2
3  # accessing the elements of a tuple using indexing
4  print("First Element of the Given Tuple:", sampleTuple[0])
5  print("Second Element of the Given Tuple:", sampleTuple[1])
6  print("Third Element of the Given Tuple:", sampleTuple[2])
7  print("Forth Element of the Given Tuple:", sampleTuple[3])
8  print("Fifth Element of the Given Tuple:", sampleTuple[4])
9  print("Sixth Element of the Given Tuple:", sampleTuple[5])
10
11 # trying to access element index more than the length of a tuple
12 try:
13     print("Seventh Element of the Given Tuple:", sampleTuple[6])
14 except Exception as e:
15     print(e)
16
17 # trying to access elements through the index of floating data type
18 try:
19     print("Accessing Second Element of the Given Tuple using floating-point index value:", sampleTuple[1.0])
20 except Exception as e:
21     print(e)
22
23 # Creating a nested tuple
24 nestedTuple = ("Fruits", [4, 6, 2, 6], (6, 2, 6, 7))
25
26 # Accessing the index of a nested tuple
27 print(nestedTuple[0][3])
28 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

```
1   sampleTuple = ("Apple", "Mango", "Banana", "Orange", "Guava", "Berries")
2
3   # Printing elements using negative indices
4   print("Element at -1 index: ", sampleTuple[-1])
5   print("Element at -2 index: ", sampleTuple[-2])
6   print("Element at -3 index: ", sampleTuple[-3])
7   print("Element at -4 index: ", sampleTuple[-4])
8   print("Element at -5 index: ", sampleTuple[-5])
9   print("Element at -6 index: ", sampleTuple[-6])
10
11  # Printing the range of elements using negative indices
12  print("Elements between -6 and -1 are: ", sampleTuple[-6:-1])
```

input

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

Slicing in Tuple

```
1   sampleTuple = ("Apple", "Mango", "Banana", "Orange", "Guava", "Berries")
2
3   # Using slicing to access elements of the tuple
4   print("Elements between indices 1 and 5: ", sampleTuple[1:5])
5
6   # Using negative indexing in slicing
7   print("Elements between indices 0 and -3: ", sampleTuple[:-3])
8
9   # Printing the entire tuple by using the default start and end values
10  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

```
1   sampleTuple = ("Apple", "Mango", "Banana", "Orange", "Guava", "Berries")
2   # printing the entire tuple for reference
3   print("Given Tuple:", sampleTuple)
4
5   # Deleting a particular element of the tuple using the del keyword
6   try:
7       del sampleTuple[3]
8       print(sampleTuple)
9   except Exception as e:
10      print(e)
11
12  # Deleting the variable from the global space of the program using the del keyword
13  del sampleTuple
14
15  # Trying accessing the tuple after deleting it
16  try:
17      print(sampleTuple)
18  except Exception as e:
19      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
1   fruits_tuple = ("mango", "orange", "banana", "apple", "papaya")
2
3   # printing the tuple before update
4   print("Before Changing the Element in Tuple...")
5   print("Tuple =", fruits_tuple)
6
7   # converting the tuple into the list
8   fruits_list = list(fruits_tuple)
9
10  # changing the element of the list
11  fruits_list[2] = "grapes"
12  print("Converting", fruits_tuple[2], "=>", fruits_list[2])
13
14  # converting the list back into the tuple
15  fruits_tuple = tuple(fruits_list)
16
17  # printing the tuple after update
18  print("After Changing the Element in Tuple...")
19  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
1   sampleTuple = (5, 3, 6, 1, 2, 8, 7, 9, 0, 4)
2
3   # printing the tuple for reference
4   print("Given Tuple =>", sampleTuple)
5
6   # using the len() method to find the length of the tuple
7   length_of_tuple = len(sampleTuple)
8
9   # printing the result for the users
10  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
8   def square( num ):
9       """
10          This function computes the square of the number.
11      """
12      return num**2
13  object_ = square(6)
14  print( "The square of the given number is: ", object_ )
```

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

Calling a Function

```python
1  def a_function( string ):
2      "This prints the value of length of string"
3      return len(string)
4
5  # Calling the function we defined
6  print( "Length of the string Functions is: ", a_function( "Functions" ) )
7  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
1  def square( item_list ):
2      '''''''This function will find the square of items in the list'''
3      squares = [ ]
4      for l in item_list:
5          squares.append( l**2 )
6      return squares
7
8  # calling the defined function
9  my_list = [17, 52, 8];
10 my_result = square( my_list )
11 print( "Squares of the list are: ", my_result )
```

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

Function Arguments

1) Default Arguments

```python
1  def function( n1, n2 = 20 ):
2      print("number 1 is: ", n1)
3      print("number 2 is: ", n2)
4
5
6  # Calling the function and passing only one argument
7  print( "Passing only one argument" )
8  function(30)
9
10 # Now giving two arguments to the function
11 print( "Passing two arguments" )
12 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
1  def function( n1, n2 ):
2      print("number 1 is: ", n1)
3      print("number 2 is: ", n2)
4
5  # Calling function and passing arguments without using keyword
6  print( "Without using keyword" )
7  function( 50, 30)
8
9  # Calling function and passing arguments using keyword
10 print( "With using keyword" )
11 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
1  def function( n1, n2 ):
2      print("number 1 is: ", n1)
3      print("number 2 is: ", n2)
4
5  # Calling function and passing two arguments out of order, we need num1 to be 20 and num2 to be 30
6  print( "Passing out of order arguments" )
7  function( 30, 20 )
8
9  # Calling function and passing only one argument
10 print( "Passing only one argument" )
11 try:
12     function( 30 )
13 except:
14     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
1  def function( *args_list ):
2      ans = []
3      for l in args_list:
4          ans.append( l.upper() )
5      return ans
6  # Passing args arguments
7  object = function('Python', 'Functions', 'tutorial')
8  print( object )
9
10 # defining a function
11 def function( **kargs_list ):
12     ans = []
13     for key, value in kargs_list.items():
14         ans.append([key, value])
15     return ans
16 # Paasing kwargs arguments
17 object = function(First = "Python", Second = "Functions", Third = "Tutorial")
18 print(object)
```

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

return Statement

```python
1  def square( num ):
2      return num**2
3
4  # Calling function and passing arguments.
5  print( "With return statement" )
6  print( square( 52 ) )
7
8  # Defining a function without return statement
9  def square( num ):
10     num**2
11
12 # Calling function and passing arguments.
13 print( "Without return statement" )
14 print( square( 52 ) )
```

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

The Anonymous Functions

```python
1  lambda_ = lambda argument1, argument2: argument1 + argument2;
2
3  # Calling the function and passing values
4  print( "Value of the function is : ", lambda_( 20, 30 ) )
5  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
1  def number( ):
2      num = 50
3      print( "Value of num inside the function: ", num)
4
5  num = 10
6  number()
7  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

```
1  def word():
2      string = 'Python functions tutorial'
3      x = 5
4      def number():
5          print( string )
6          print( x )
7
8      number()
9  word()
```

```
Python functions tutorial
5
```

Python Built-in Functions

Python abs() Function

```
1  integer = -20
2  print('Absolute value of -40 is:', abs(integer))
3
4  #  floating number
5  floating = -20.83
6  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

```
1   k = [1, 3, 4, 6]
2   print(all(k))
3
4   # all values false
5   k = [0, False]
6   print(all(k))
7
8   # one false value
9   k = [1, 3, 7, 0]
10  print(all(k))
11
12  # one true value
13  k = [0, False, 5]
14  print(all(k))
15
16  # empty iterable
17  k = []
18  print(all(k))
```

```
True
False
False
False
True
```

Python bin() Function

```
1  x =  10
2  y =  bin(x)
3  print (y)
```

```
0b1010
```

Python bool()

```
1   test1 = []
2   print(test1,'is',bool(test1))
3   test1 = [0]
4   print(test1,'is',bool(test1))
5   test1 = 0.0
6   print(test1,'is',bool(test1))
7   test1 = None
8   print(test1,'is',bool(test1))
9   test1 = True
10  print(test1,'is',bool(test1))
11  test1 = 'Easy string'
12  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

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

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

Python exec() Function

```
1  x = 8
2  exec('print(x==8)')
3  exec('print(x+4)')
```

```
True
12
```

Python any() Function

```
1  l = [4, 3, 2, 0]
2  print(any(l))
3
4  l = [0, False]
5  print(any(l))
6
7  l = [0, False, 5]
8  print(any(l))
9
10 l = []
11 print(any(l))
```

```
True
False
True
False
```

Python float()

```
1  # for integers
2  print(float(9))
3
4  # for floats
5  print(float(8.19))
6
7  # for string floats
8  print(float("-24.27"))
9
10 # for string floats with whitespaces
11 print(float("    -17.19\n"))
12
13 # string float error
14 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

```
1   # d, f and b are a type
2
3   # integer
4   print(format(123, "d"))
5
6   # float arguments
7   print(format(123.4567898, "f"))
8
9   # binary format
10  print(format(12, "b"))
```

```
123
123.456790
1100
```

Python hasattr() Function

```
1   l = [4, 3, 2, 0]
2   print(any(l))
3
4   l = [0, False]
5   print(any(l))
6
7   l = [0, False, 5]
8   print(any(l))
9
10  l = []
11  print(any(l))
```

```
True
False
True
False
```

Python iter() Function Example

```
 1  # list of numbers
 2  list = [1,2,3,4,5]
 3
 4  listIter = iter(list)
 5
 6  # prints '1'
 7  print(next(listIter))
 8
 9  # prints '2'
10  print(next(listIter))
11
12  # prints '3'
13  print(next(listIter))
14
15  # prints '4'
16  print(next(listIter))
17
18  # prints '5'
19  print(next(listIter))
```

```
1
2
3
4
5
```

Python len() Function

```
1  strA = 'Python'
2  print(len(strA))
```

```
6
```

Python list()

```
 1  # empty list
 2  print(list())
 3
 4  # string
 5  String = 'abcde'
 6  print(list(String))
 7
 8  # tuple
 9  Tuple = (1,2,3,4,5)
10  print(list(Tuple))
11  # list
12  List = [1,2,3,4,5]
13  print(list(List))
```

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

Python memoryview() Function

```python
1   #A random bytearray
2   randomByteArray = bytearray('ABC', 'utf-8')
3
4   mv = memoryview(randomByteArray)
5
6   # access the memory view's zeroth index
7   print(mv[0])
8
9   # It create byte from memory view
10  print(bytes(mv[0:2]))
11
12  # It create list from memory view
13  print(list(mv[0:3]))
```

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

Python object()

```python
1   python = object()
2
3   print(type(python))
4   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
1   # Calling function
2   result = hash(21) # integer value
3   result2 = hash(22.2) # decimal value
4   # Displaying result
5   print(result)
6   print(result2)
```

```
21
461168601842737174
```

Python pow() Function

```python
1   # positive x, positive y (x**y)
2   print(pow(4, 2))
3
4   # negative x, positive y
5   print(pow(-4, 2))
6
7   # positive x, negative y (x**-y)
8   print(pow(4, -2))
9
10  # negative x, negative y
11  print(pow(-4, -2))
```

```
16
16
0.0625
0.0625
```

Python reversed() Function

```
1   # for string
2   String = 'Java'
3   print(list(reversed(String)))
4
5   # for tuple
6   Tuple = ('J', 'a', 'v', 'a')
7   print(list(reversed(Tuple)))
8
9   # for range
10  Range = range(8, 12)
11  print(list(reversed(Range)))
12
13  # for list
14  List = [1, 2, 7, 5]
15  print(list(reversed(List)))
```

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

Python issubclass() Function

```
1   class Rectangle:
2     def __init__(rectangleType):
3       print('Rectangle is a ', rectangleType)
4
5   class Square(Rectangle):
6     def __init__(self):
7       Rectangle.__init__('square')
8
9   print(issubclass(Square, Rectangle))
10  print(issubclass(Square, list))
11  print(issubclass(Square, (list, Rectangle)))
12  print(issubclass(Rectangle, (list, Rectangle)))
```

```
True
False
True
True
```

Python tuple() Function

```
1   t1 = tuple()
2   print('t1=', t1)
3
4   # creating a tuple from a list
5   t2 = tuple([1, 6, 9])
6   print('t2=', t2)
7
8   # creating a tuple from a string
9   t1 = tuple('Java')
10  print('t1=',t1)
11
12  # creating a tuple from a dictionary
13  t1 = tuple({4: 'four', 5: 'five'})
14  print('t1=',t1)
```

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

Python zip() Function

```
1   numList = [4,5, 6]
2   strList = ['four', 'five', 'six']
3
4   # No iterables are passed
5   result = zip()
6
7   # Converting itertor to list
8   resultList = list(result)
9   print(resultList)
10
11  # Two iterables are passed
12  result = zip(numList, strList)
13
14  # Converting itertor to set
15  resultSet = set(result)
16  print(resultSet)
```

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

Python Lambda Functions

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

```
10
```

```
1 def add( num ):
2     return num + 4
3 print( add(6) )
```
10

```
1 a = lambda x, y : (x * y)
2 print(a(4, 5))
```
10

```
1 a = lambda x, y, z : (x + y + z)
2 print(a(4, 5, 5))
```
14

What's the Distinction Between Lambda and Def Functions?

```
1 def reciprocal( num ):
2     return 1 / num
3
4 lambda_reciprocal = lambda num: 1 / num
5
6 # using the function defined by def keyword
7 print( "Def keyword: ", reciprocal(6) )
8
9 # using the function defined by lambda keyword
10 print( "Lambda keyword: ", lambda_reciprocal(6) )
```
14

Using Lambda Function with filter()

```
1 list_ = [35, 12, 69, 55, 75, 14, 73]
2 odd_list = list(filter( lambda num: (num % 2 != 0) , list_ ))
3 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()

```
1  numbers_list = [2, 4, 5, 1, 3, 7, 8, 9, 10]
2  squared_list = list(map( lambda num: num ** 2 , numbers_list ))
3  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

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

input

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

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

The greater number is: 35

Using Lambda with Multiple Statements

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

input

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