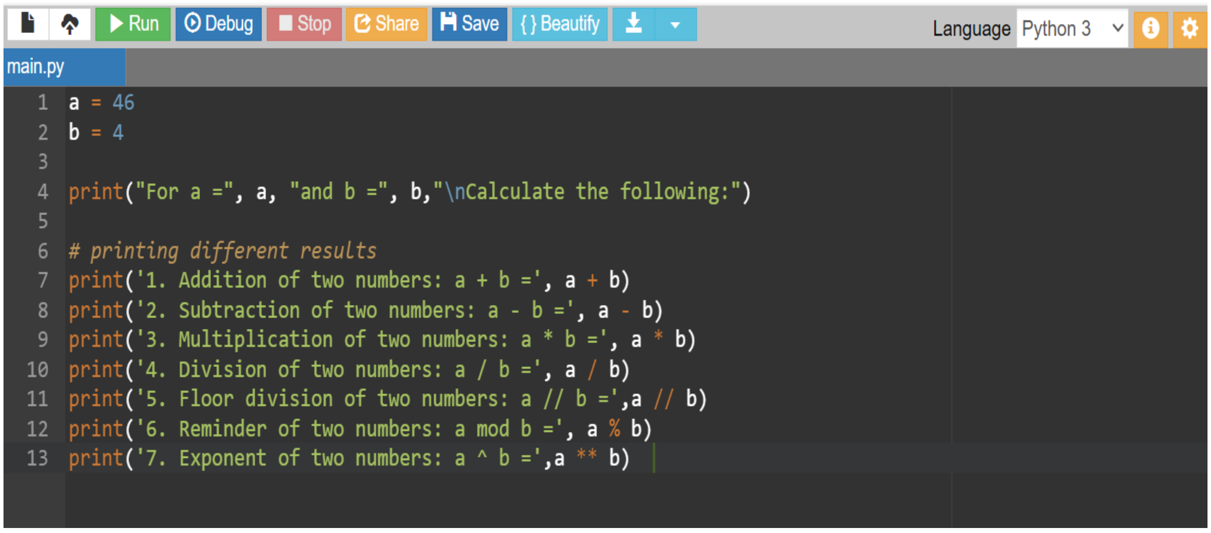
PYTHON ASSIGNEMT

The **Operators** are the symbols used to perform a specific operation on different values and variables. These values and variables are considered as the **Operands**, on which the operator is applied.

Different Types of Operators in Python

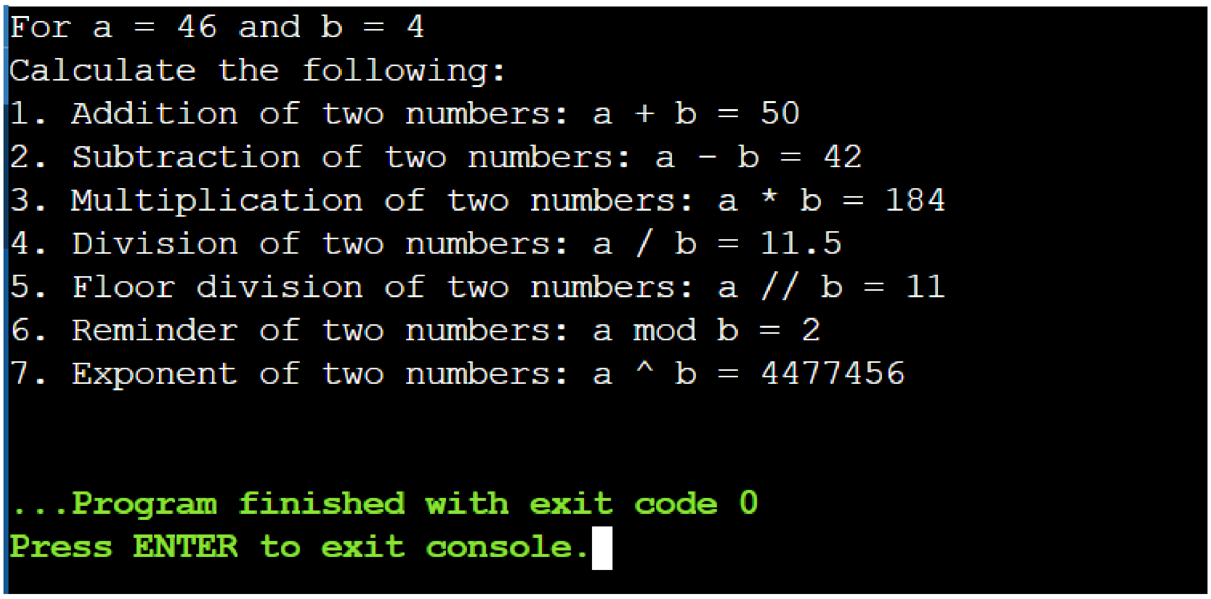
1. **Arithmetic Operators**

CODE



```python
a = 46
b = 4

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)
```

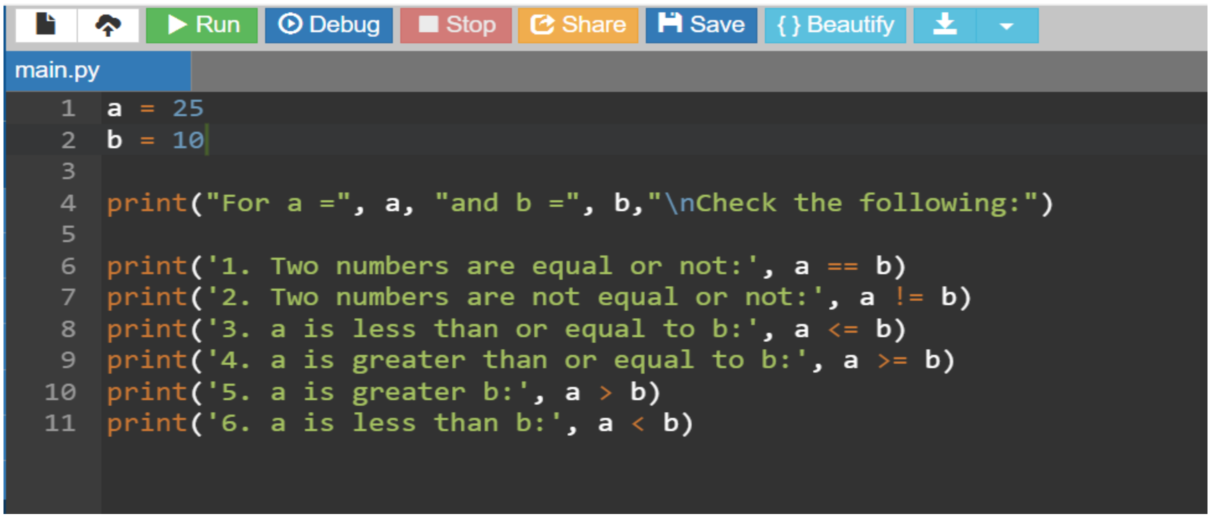
OUTPUT



```
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


...Program finished with exit code 0
Press ENTER to exit console.
```

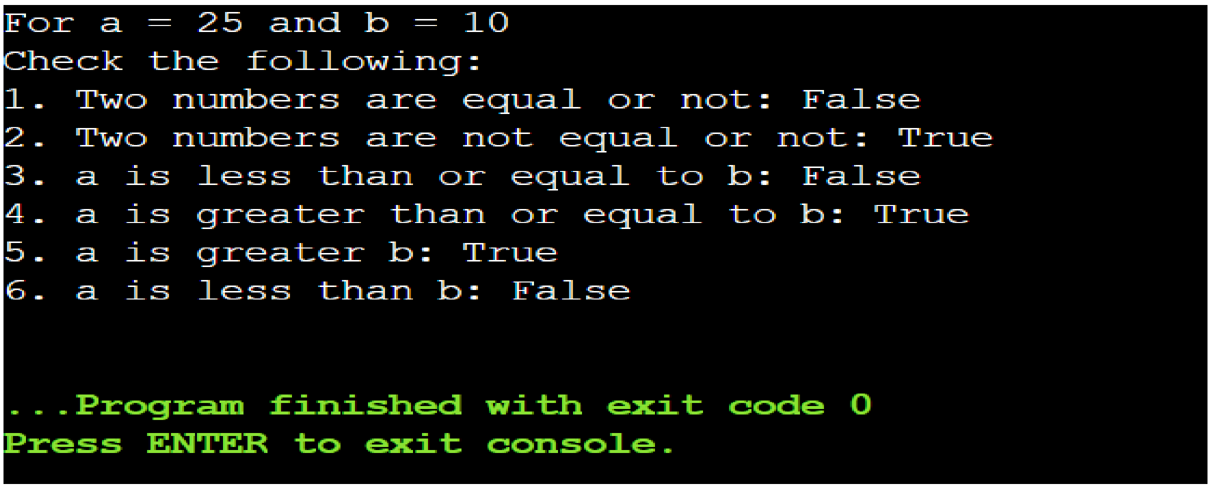
2. Comparison Operators

CODE



```python
a = 25
b = 10

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)
```

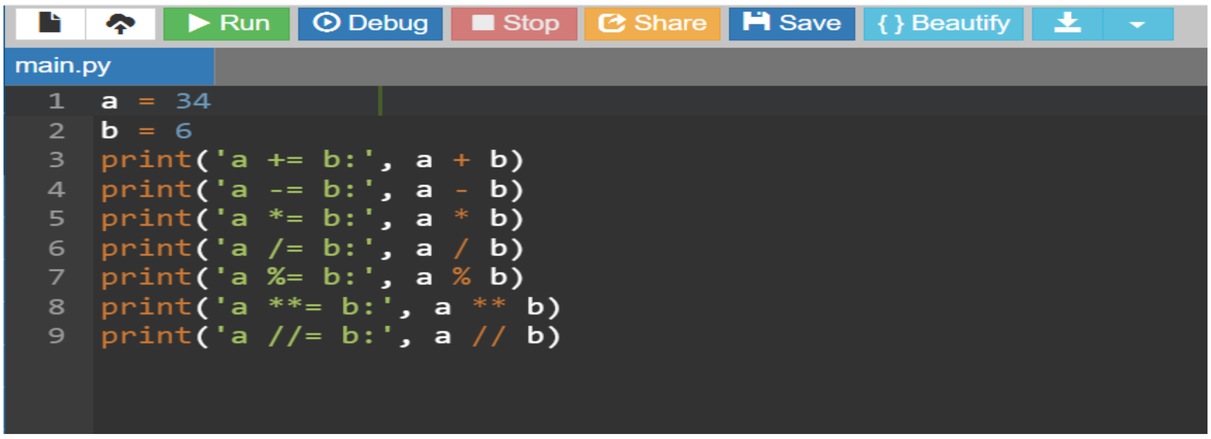
OUTPUT



```
For a = 25 and b = 10
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


...Program finished with exit code 0
Press ENTER to exit console.
```

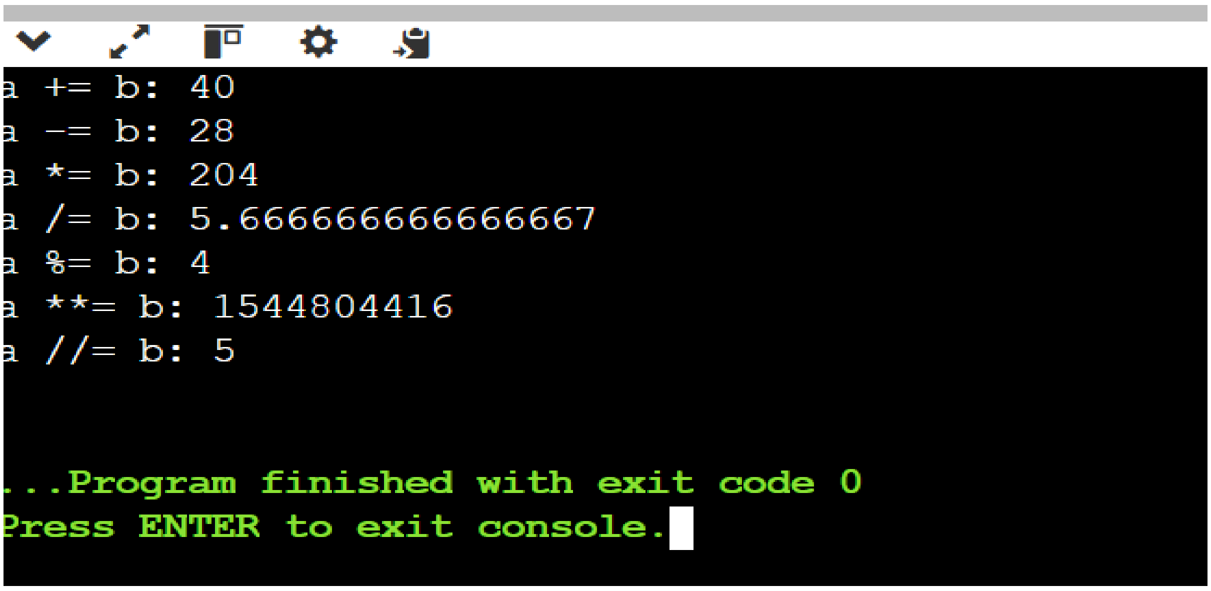
3. Assignment Operators

CODE



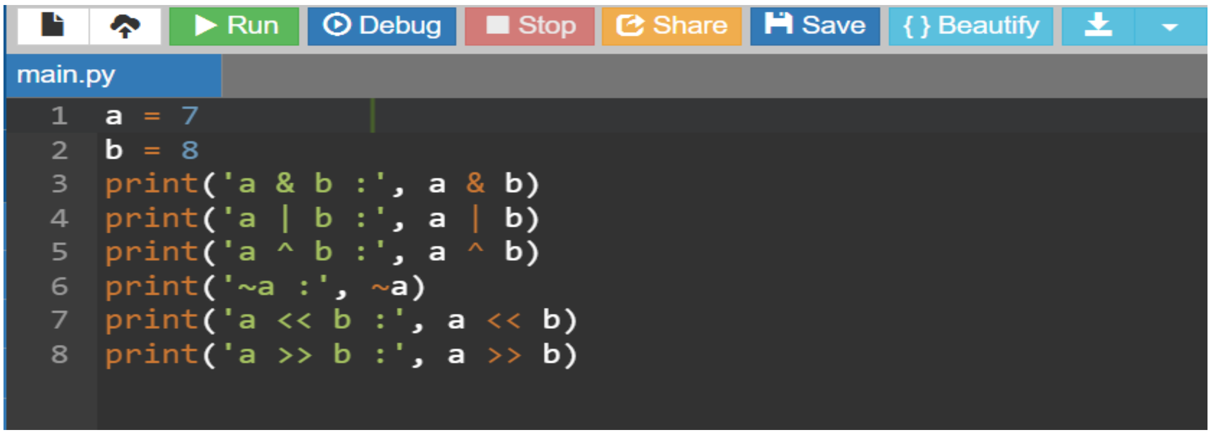
```python
a = 34
b = 6
print('a += b:', a + b)
print('a -= b:', a - b)
print('a *= b:', a * b)
print('a /= b:', a / b)
print('a %= b:', a % b)
print('a **= b:', a ** b)
print('a //= b:', a // b)
```

OUTPUT


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


...Program finished with exit code 0
Press ENTER to exit console.
```

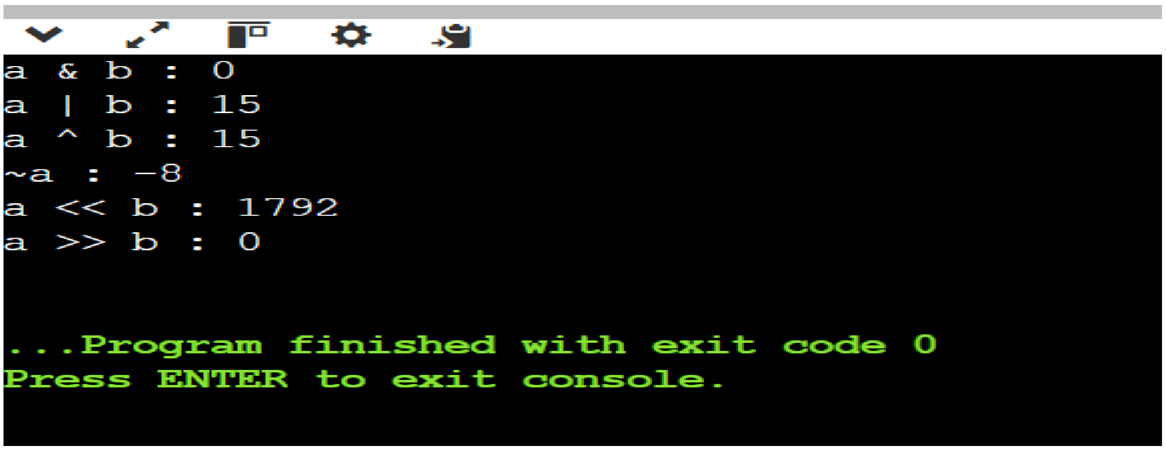
4. Logical Operators

CODE



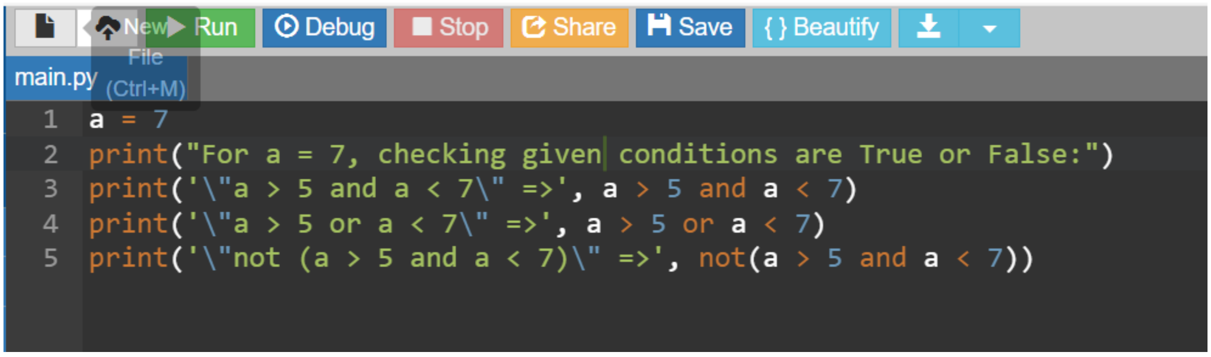
```python
a = 7
b = 8
print('a & b :', a & b)
print('a | b :', a | b)
print('a ^ b :', a ^ b)
print('~a :', ~a)
print('a << b :', a << b)
print('a >> b :', a >> b)
```

OUTPUT


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


...Program finished with exit code 0
Press ENTER to exit console.
```

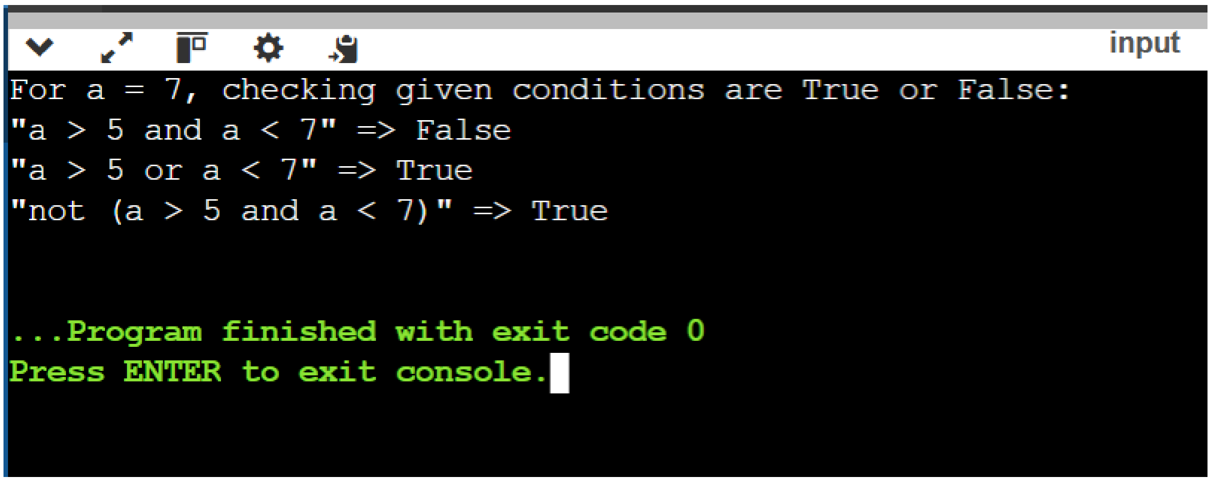
5. **Bitwise Operators**

CODE



```python
a = 7
print("For a = 7, checking given 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))
```

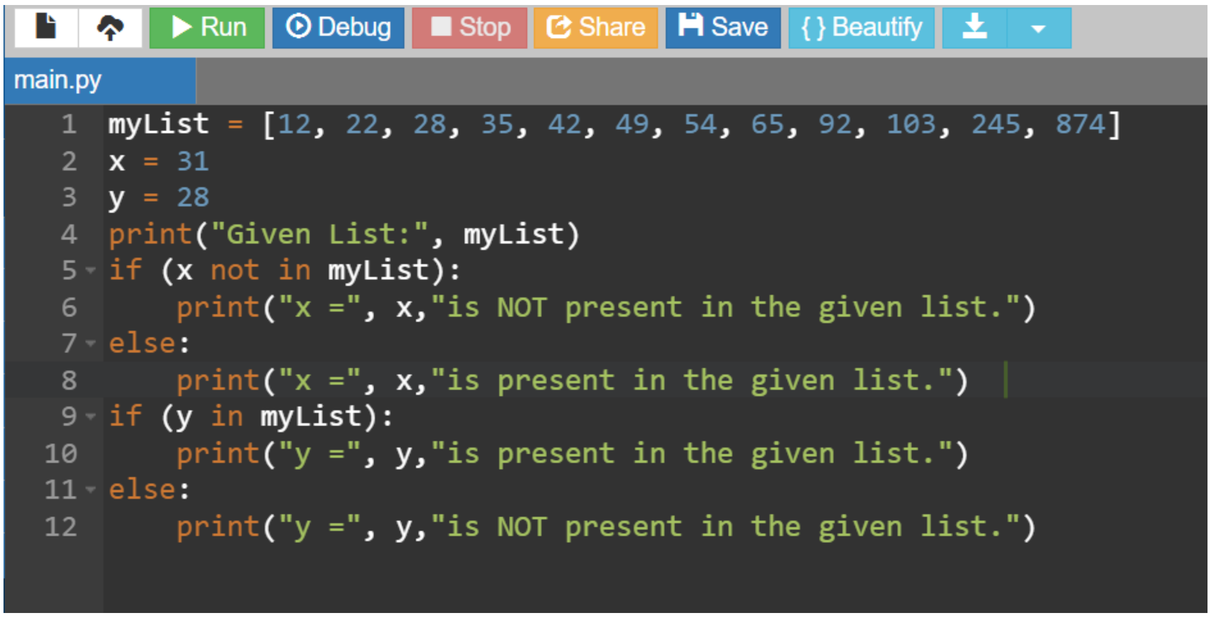
OUTPUT



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


...Program finished with exit code 0
Press ENTER to exit console.
```

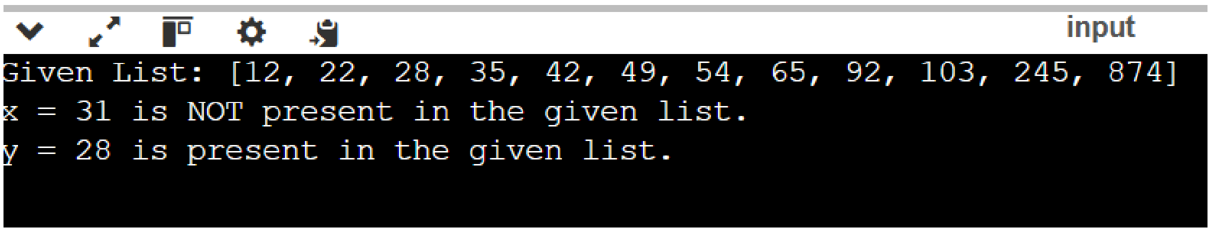
6. **Membership Operators**

CODE



```python
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.")
```

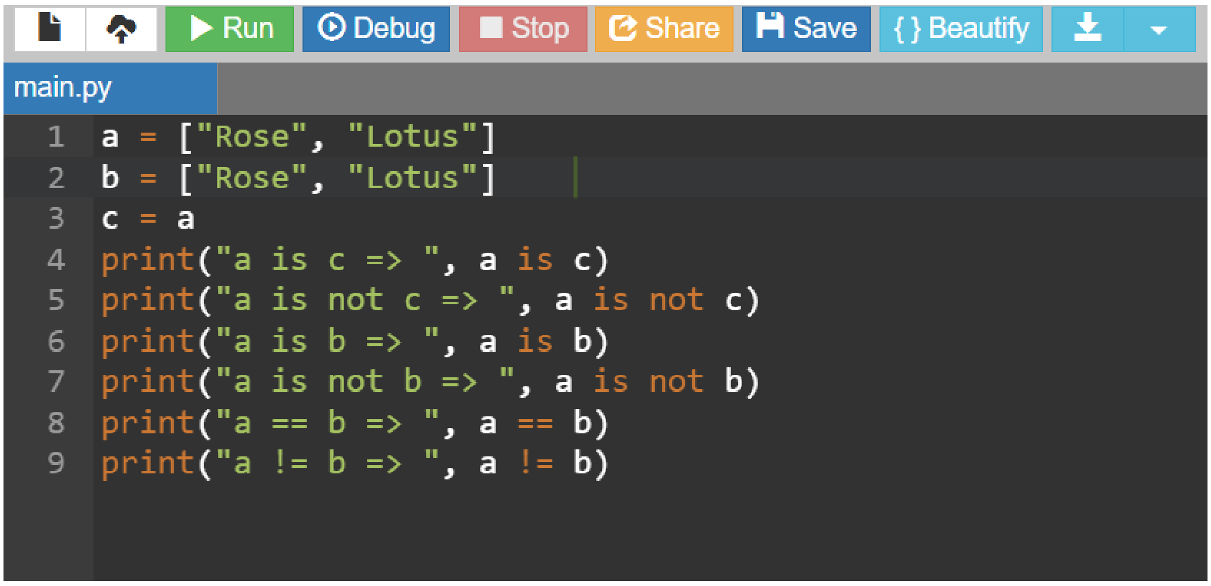
OUTPUT



```
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.
```

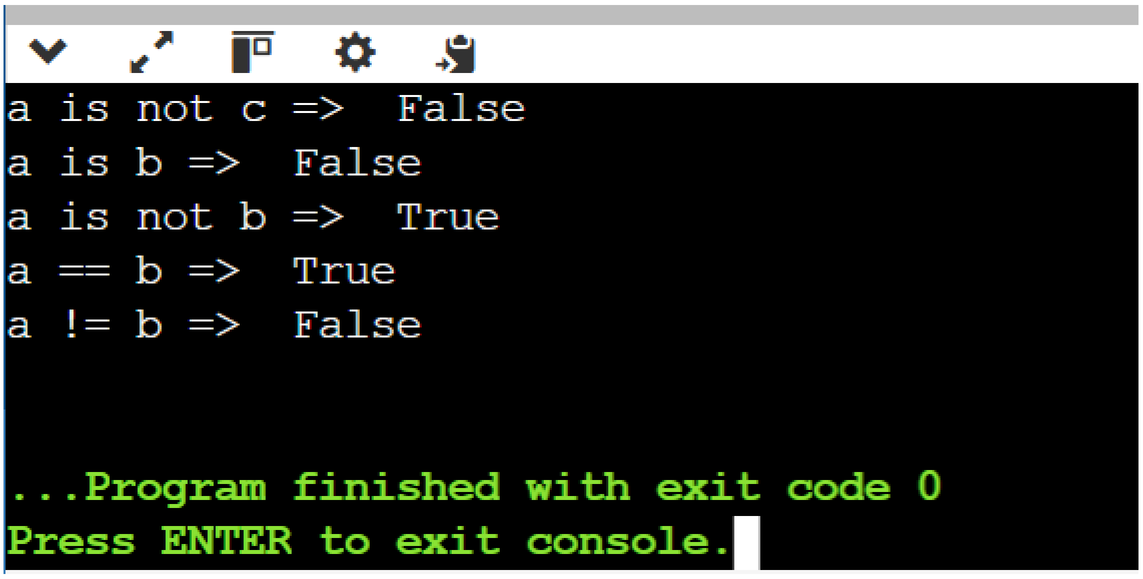
7. **Identity Operators**

   CODE



```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)
```

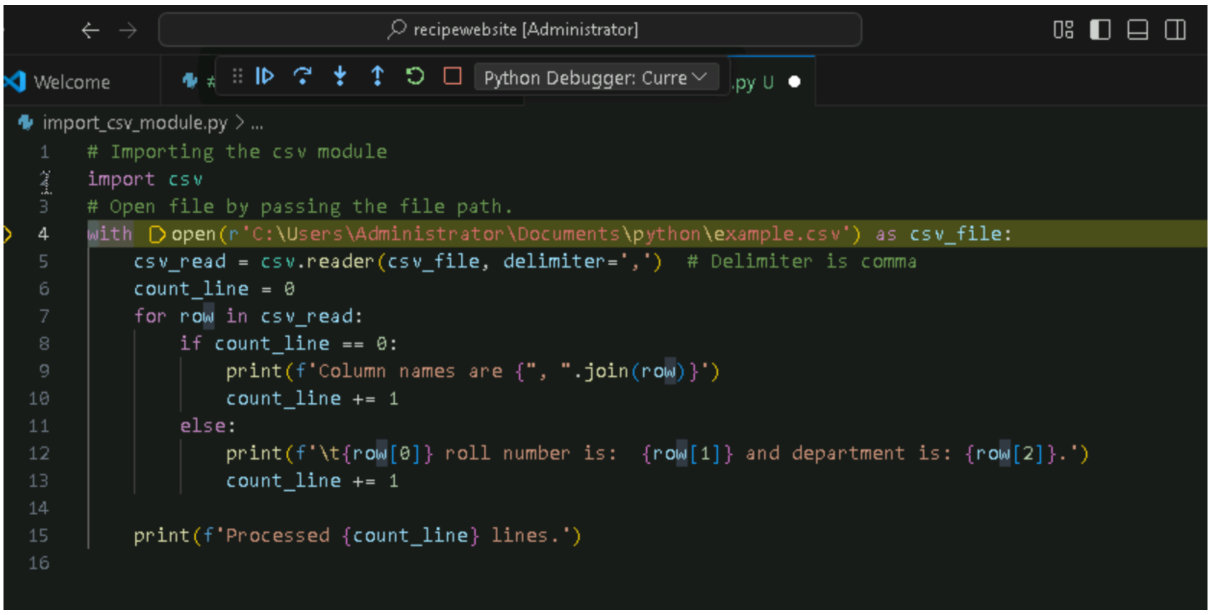
OUTPUT



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


...Program finished with exit code 0
Press ENTER to exit console.
```

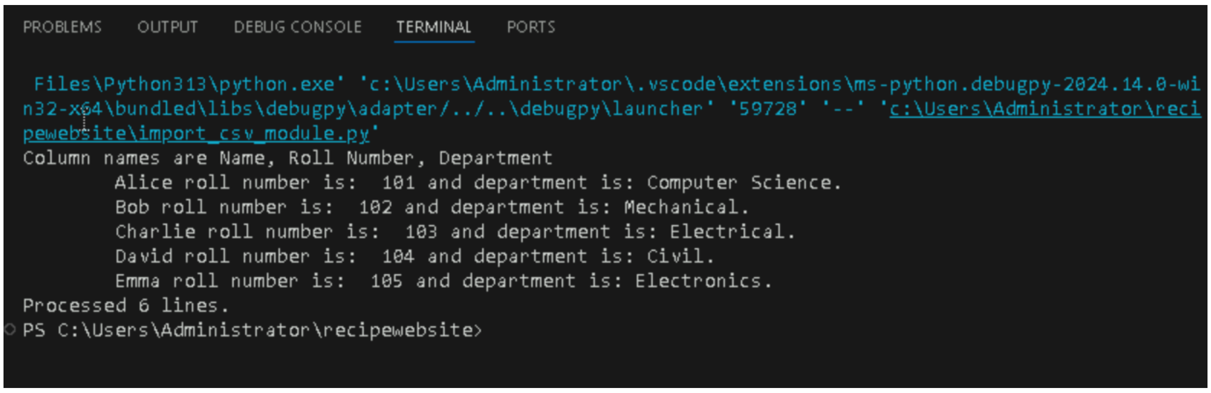
To Read CSV file in Python

CODE



```
# Importing the csv module
import csv
# Open file by passing the file path.
with open(r'C:\Users\Administrator\Documents\python\example.csv') as csv_file:
    csv_read = csv.reader(csv_file, delimiter=',')  # Delimiter is comma
    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.')
```

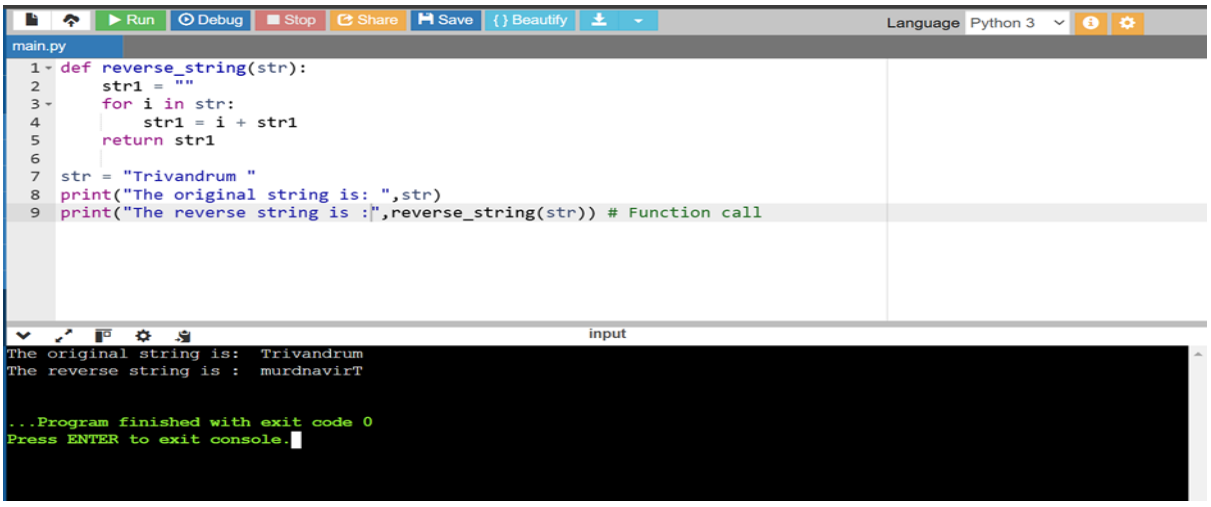
OUTPUT



```
 Files\Python313\python.exe' 'c:\Users\Administrator\.vscode\extensions\ms-python.debugpy-2024.14.0-wi
n32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '59728' '--' 'c:\Users\Administrator\reci
pewebsite\import_csv_module.py'
Column names are Name, Roll Number, Department
        Alice roll number is:  101 and department is: Computer Science.
        Bob roll number is:  102 and department is: Mechanical.
        Charlie roll number is:  103 and department is: Electrical.
        David roll number is:  104 and department is: Civil.
        Emma roll number is:  105 and department is: Electronics.
Processed 6 lines.
PS C:\Users\Administrator\recipewebsite>
```

REVERSE A STRING

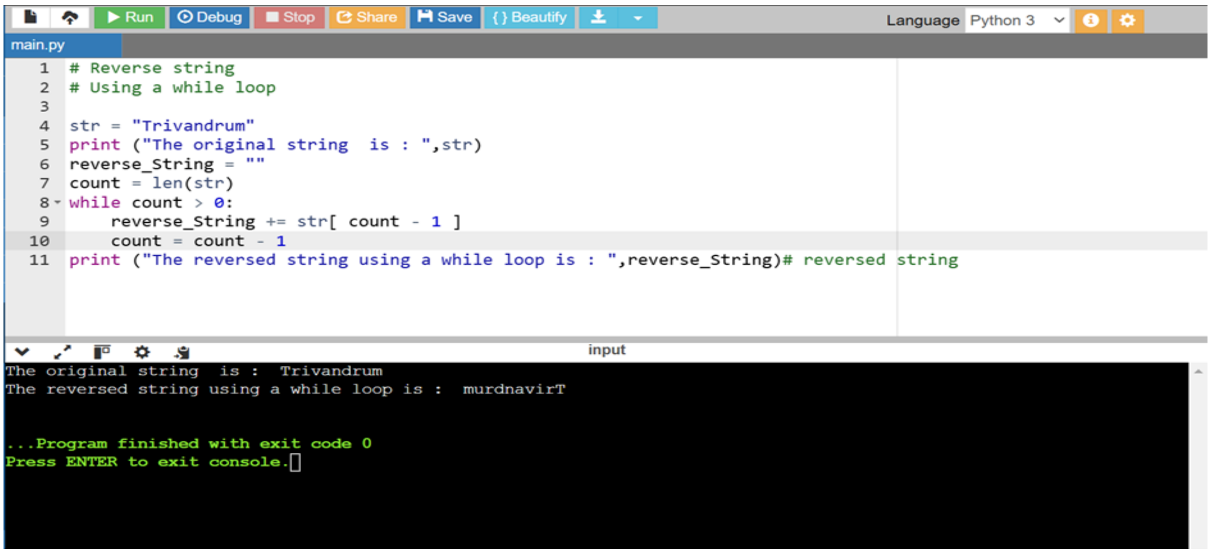
1. Using FOR Loop



```
def reverse_string(str):
    str1 = ""
    for i in str:
        str1 = i + str1
    return str1

str = "Trivandrum "
print("The original string is: ",str)
print("The reverse string is :",reverse_string(str)) # Function call
```

```
The original string is:  Trivandrum
The reverse string is :  murdnavirT


...Program finished with exit code 0
Press ENTER to exit console.
```

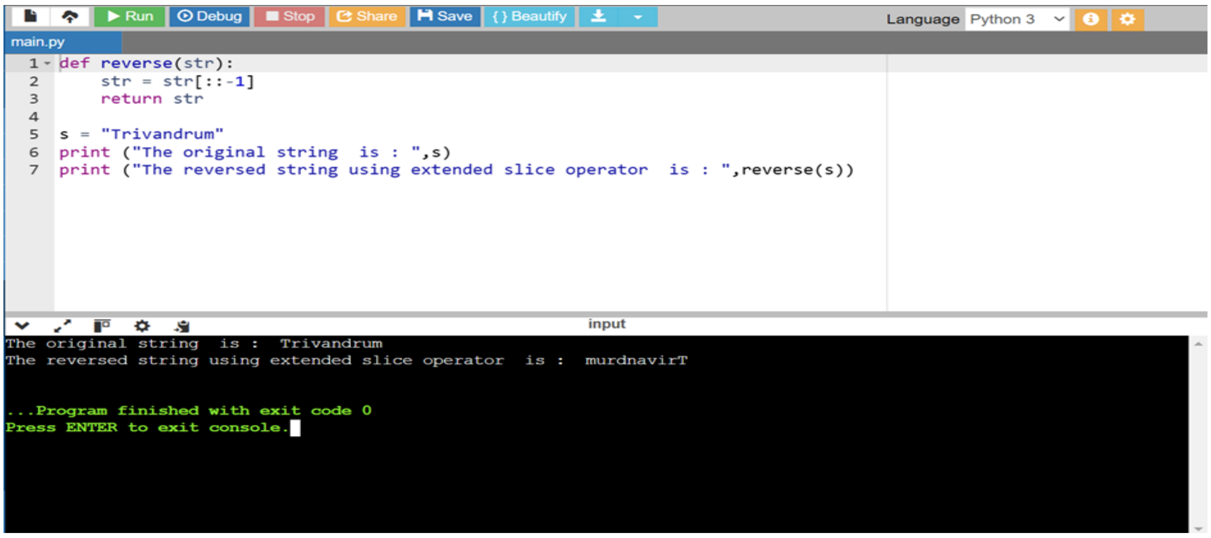
2. Using WHILE Loop



```python
# Reverse string
# Using a while loop

str = "Trivandrum"
print ("The original string  is : ",str)
reverse_String = ""
count = len(str)
while count > 0:
    reverse_String += str[ count - 1 ]
    count = count - 1
print ("The reversed string using a while loop is : ",reverse_String)# reversed string
```

```
The original string  is :  Trivandrum
The reversed string using a while loop is :  murdnavirT


...Program finished with exit code 0
Press ENTER to exit console.
```

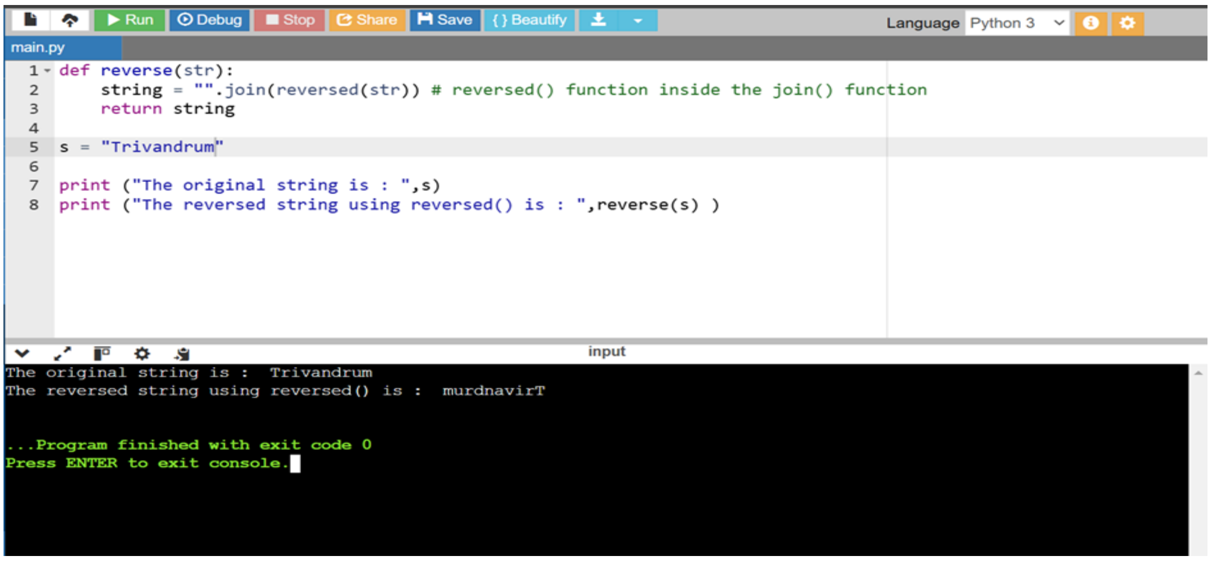
3. Using the slice operator



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

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

```
The original string  is :  Trivandrum
The reversed string using extended slice operator  is :  murdnavirT


...Program finished with exit code 0
Press ENTER to exit console.
```

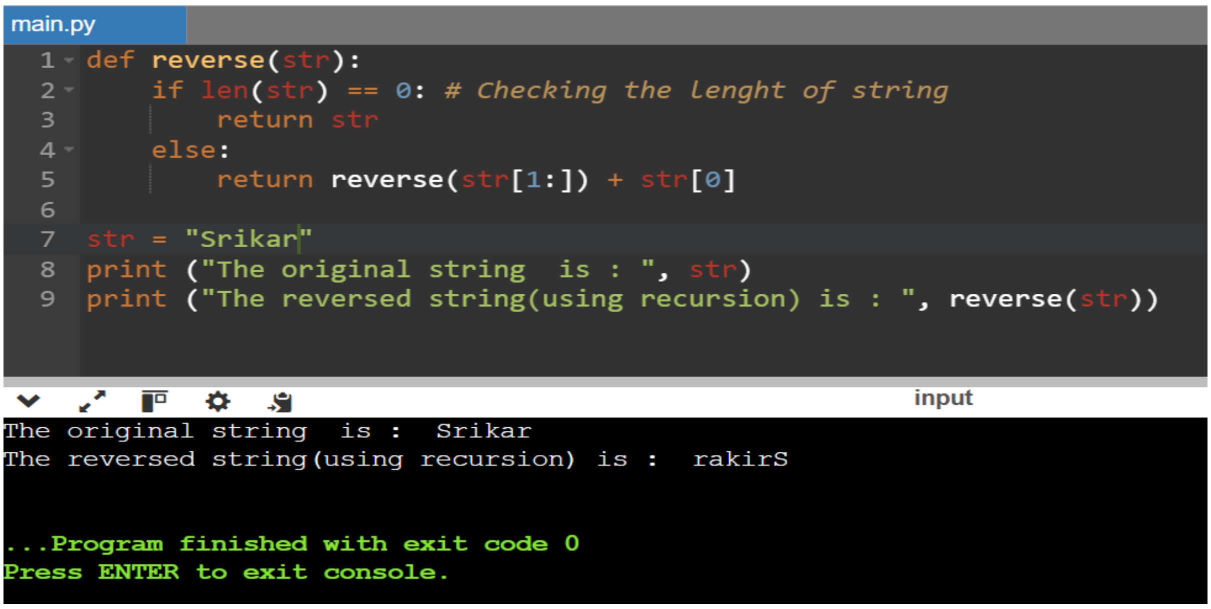
4. Using the reverse () function



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

s = "Trivandrum"

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

```
The original string is :  Trivandrum
The reversed string using reversed() is :  murdnavirT


...Program finished with exit code 0
Press ENTER to exit console.
```

5. Using the Recursion

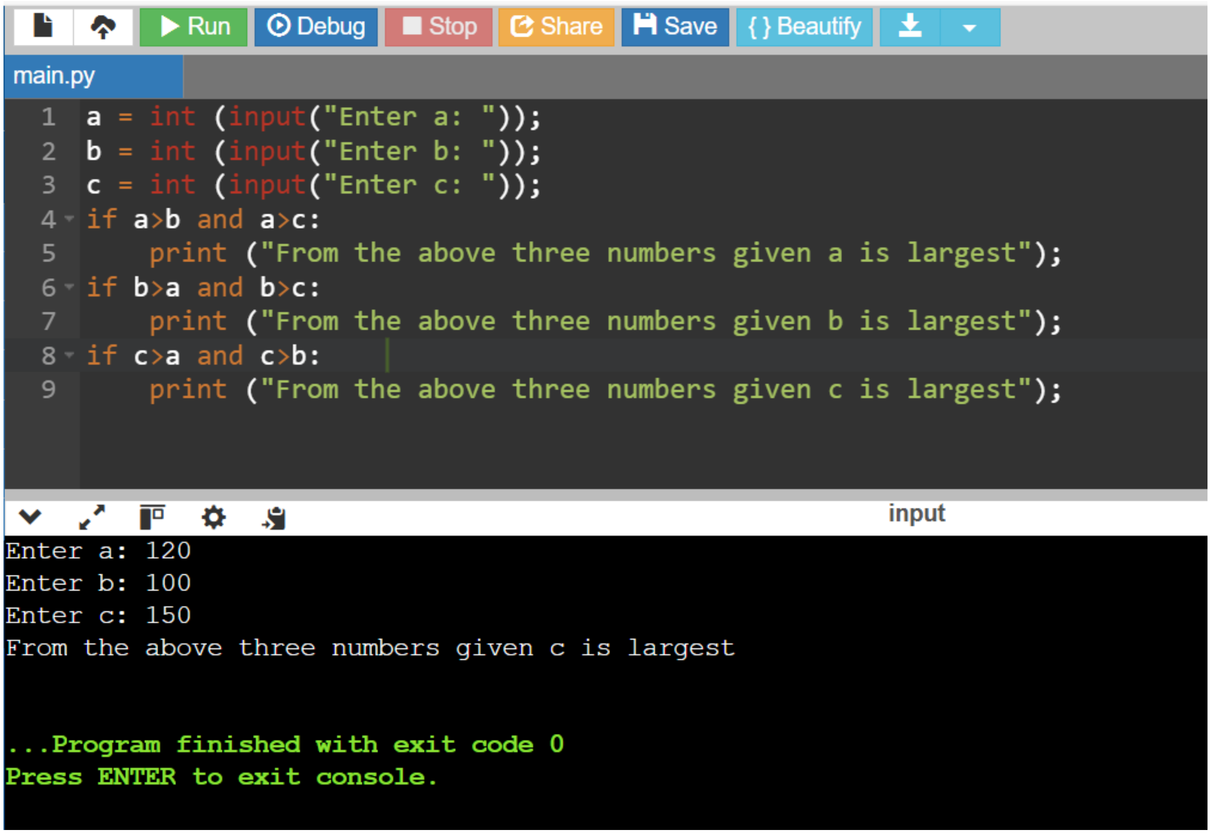


```python
def reverse(str):
    if len(str) == 0: # Checking the Lenght of string
        return str
    else:
        return reverse(str[1:]) + str[0]

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

```
The original string  is :  Srikar
The reversed string(using recursion) is :  rakirS


...Program finished with exit code 0
Press ENTER to exit console.
```

If Statement:

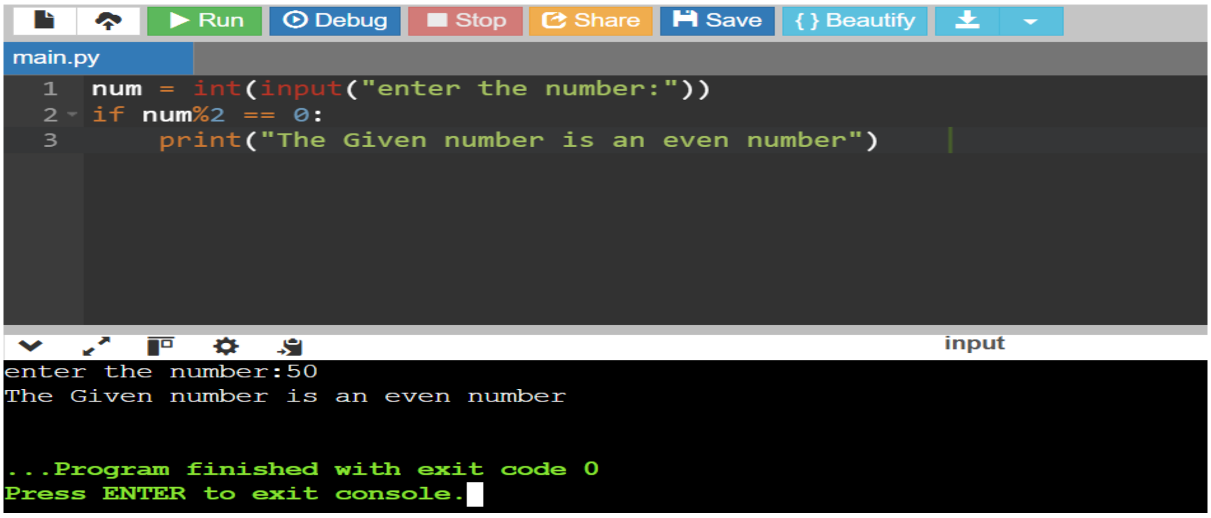
Example 1:



```python
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: 120
Enter b: 100
Enter c: 150
From the above three numbers given c is largest


...Program finished with exit code 0
Press ENTER to exit console.
```

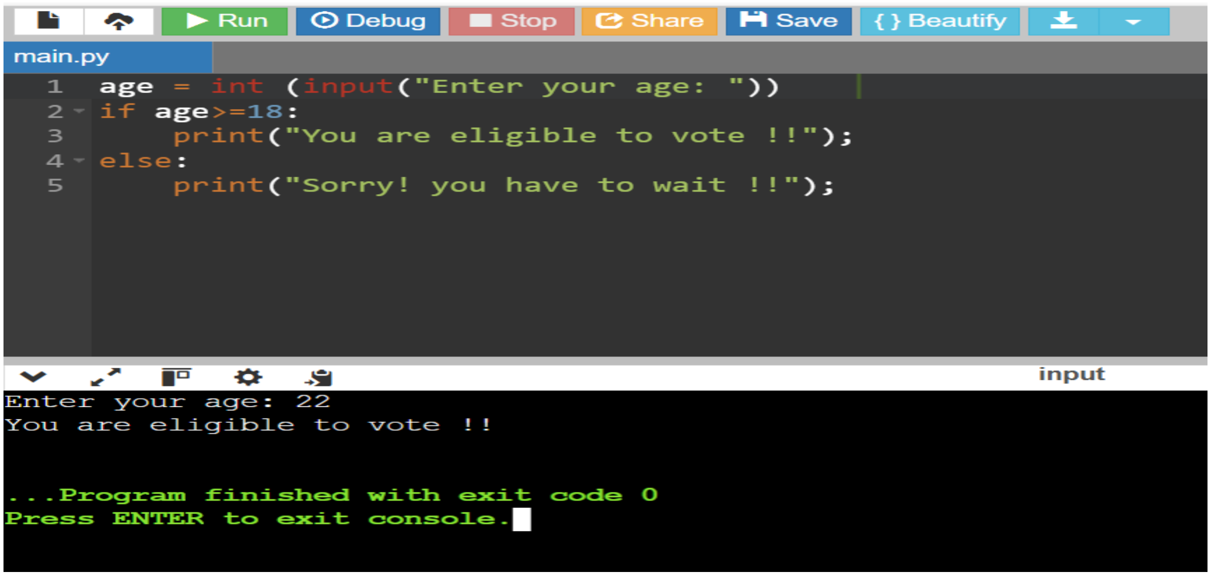
Example 2:



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

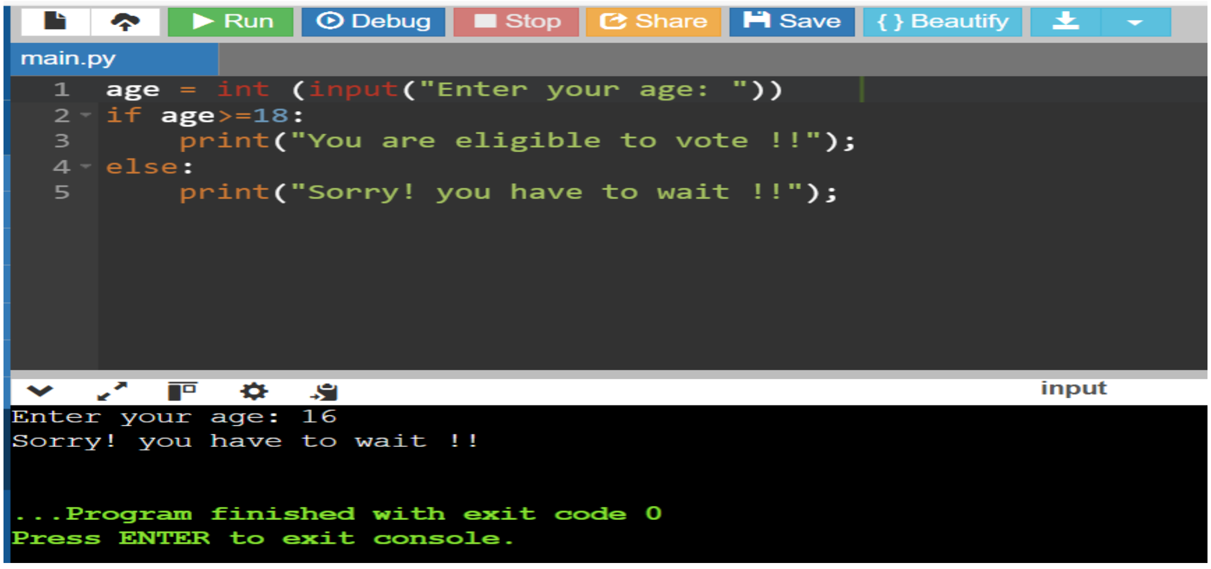
```
enter the number:50
The Given number is an even number

...Program finished with exit code 0
Press ENTER to exit console.
```

If-Else Statement:



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

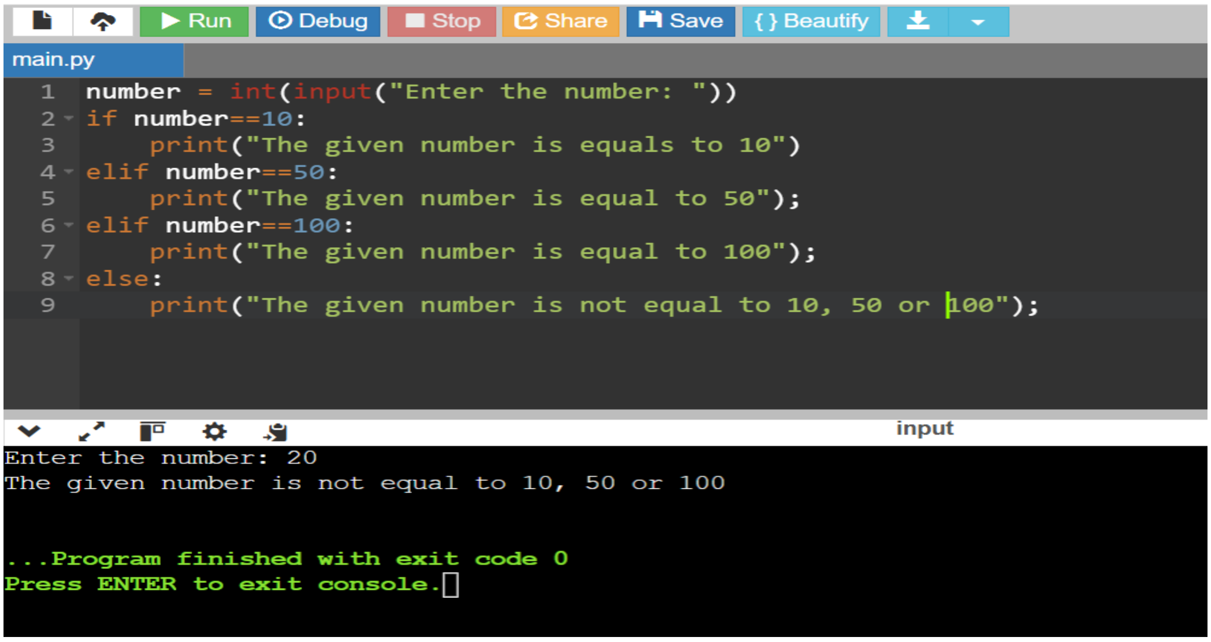
```
Enter your age: 22
You are eligible to vote !!

...Program finished with exit code 0
Press ENTER to exit console.
```



```python
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: 16
Sorry! you have to wait !!

...Program finished with exit code 0
Press ENTER to exit console.
```

Elif Statement:

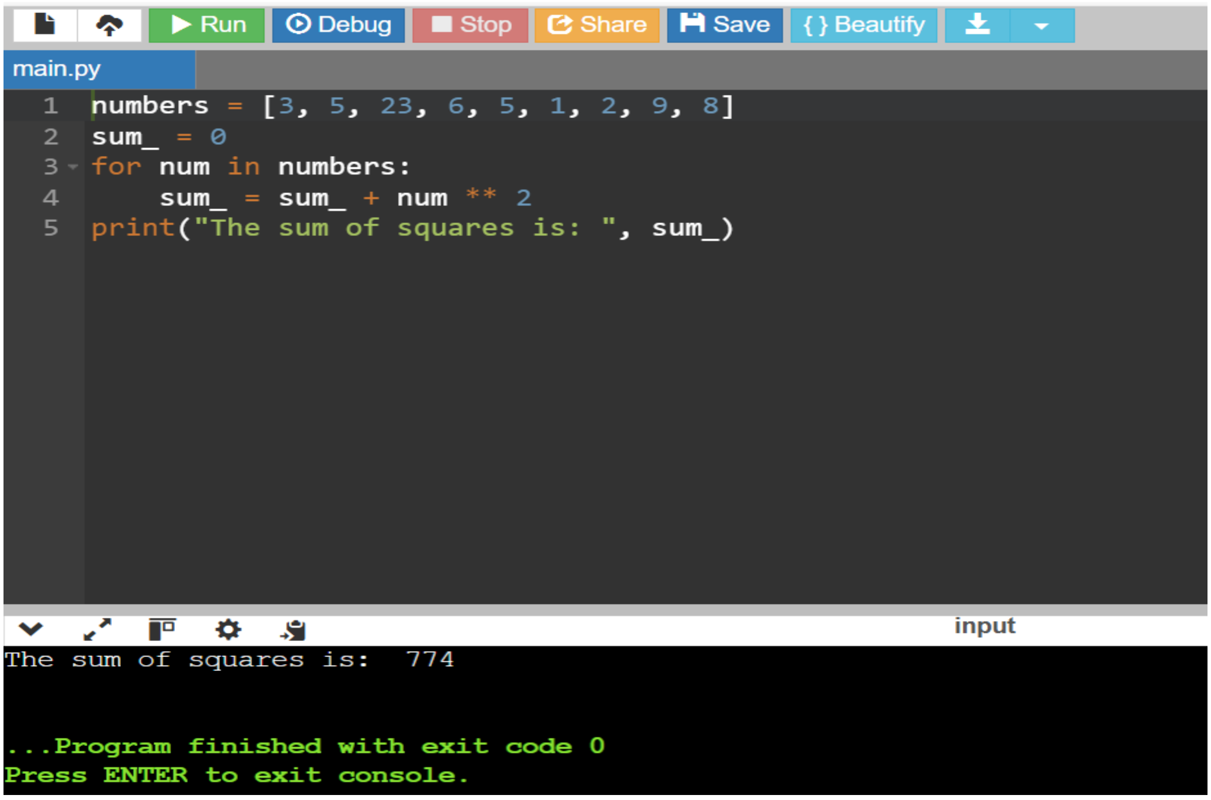


```
number = int(input("Enter the number: "))
if number==10:
    print("The given number is equals to 10")
elif number==50:
    print("The given number is equal to 50");
elif number==100:
    print("The given number is equal to 100");
else:
    print("The given number is not equal to 10, 50 or 100");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

```
Enter the number: 20
The given number is not equal to 10, 50 or 100


...Program finished with exit code 0
Press ENTER to exit console.
```

FOR Loops:

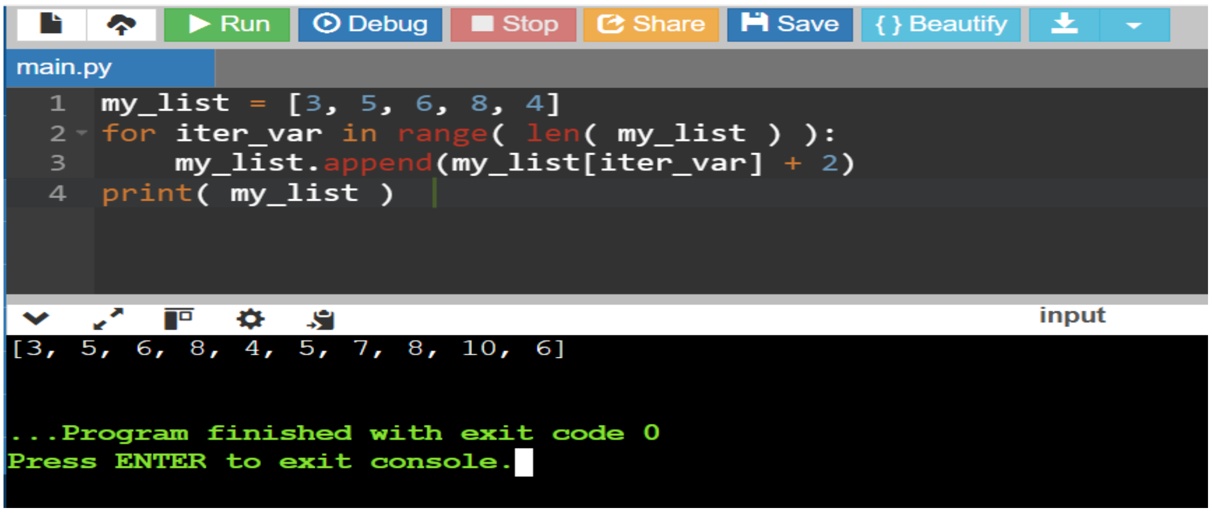
1. **Iterating by using index of sequence**



```
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


...Program finished with exit code 0
Press ENTER to exit console.
```

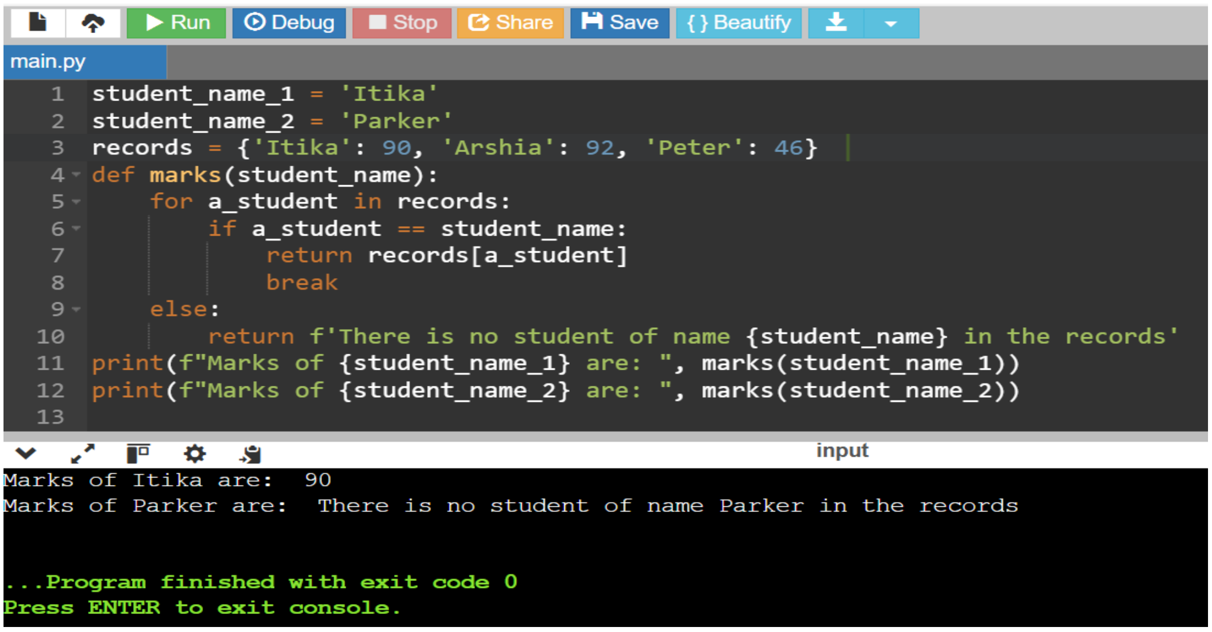
2. **Using Range ()**



```
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]


...Program finished with exit code 0
Press ENTER to exit console.
```

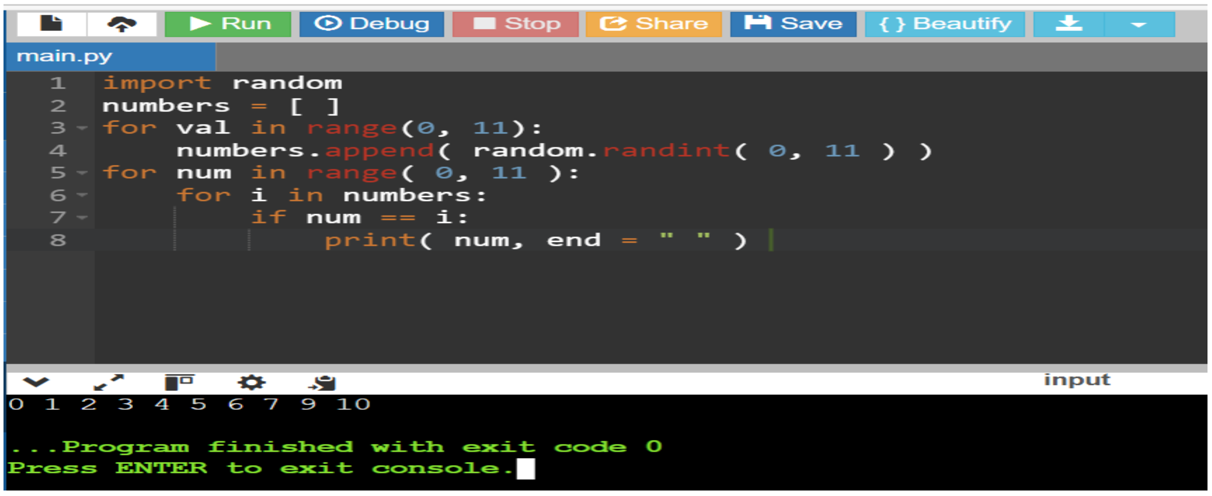
3. **Using else statement with loop**



```
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))
```

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


...Program finished with exit code 0
Press ENTER to exit console.
```

4. **Nested loop**

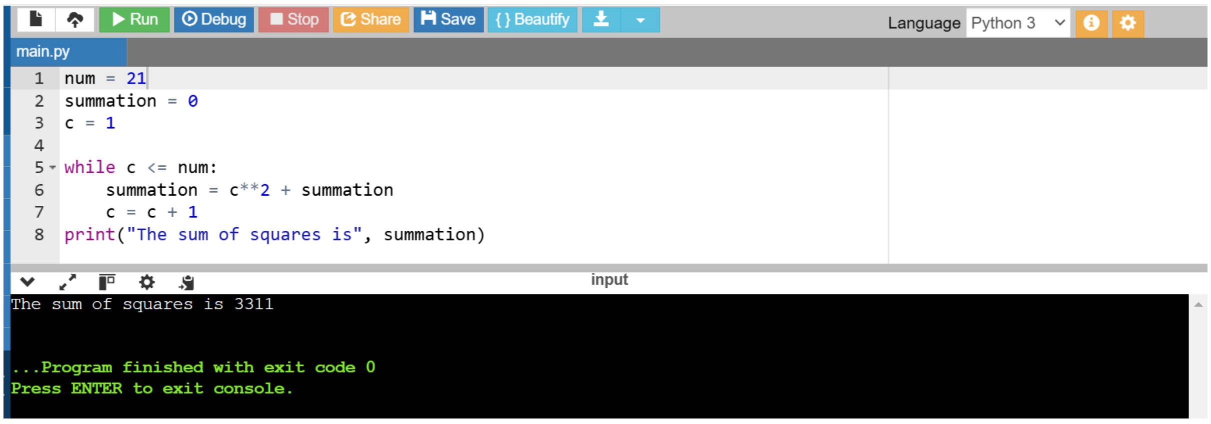


```
import random
numbers = [ ]
for val in range(0, 11):
    numbers.append( random.randint( 0, 11 ) )
for num in range( 0, 11 ):
    for i in numbers:
        if num == i:
            print( num, end = " " )
```

```
0 1 2 3 4 5 6 7 9 10

...Program finished with exit code 0
Press ENTER to exit console.
```

WHILE Loops:

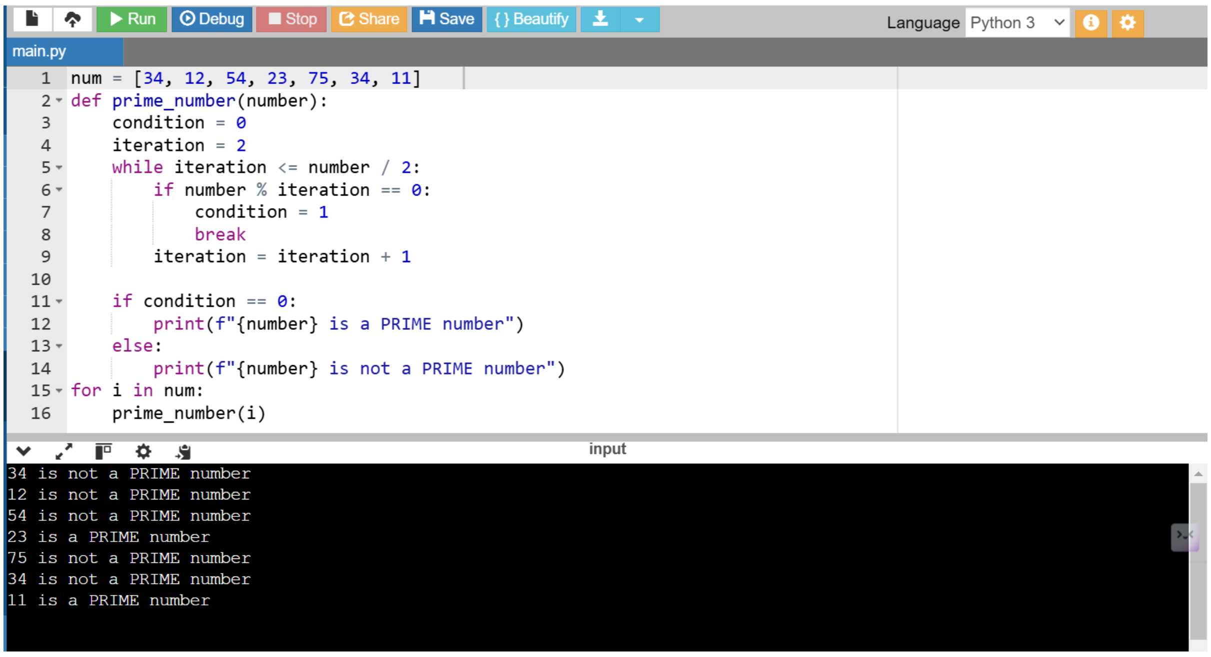
1. Sum of squares



```python
num = 21
summation = 0
c = 1

while c <= num:
    summation = c**2 + summation
    c = c + 1
print("The sum of squares is", summation)
```

```
The sum of squares is 3311


...Program finished with exit code 0
Press ENTER to exit console.
```

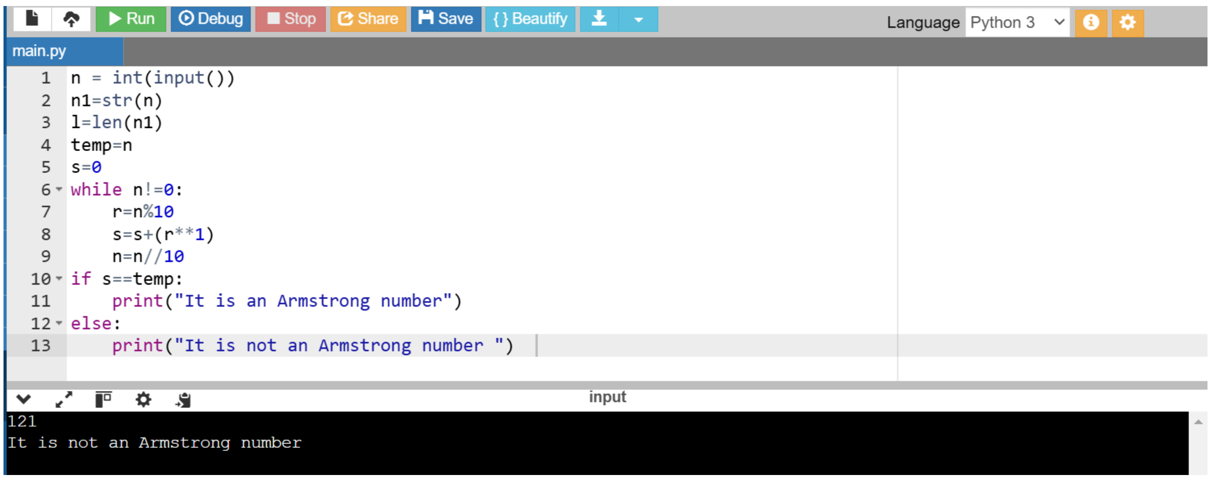
2. To check whether given number is Prime or not



```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
```

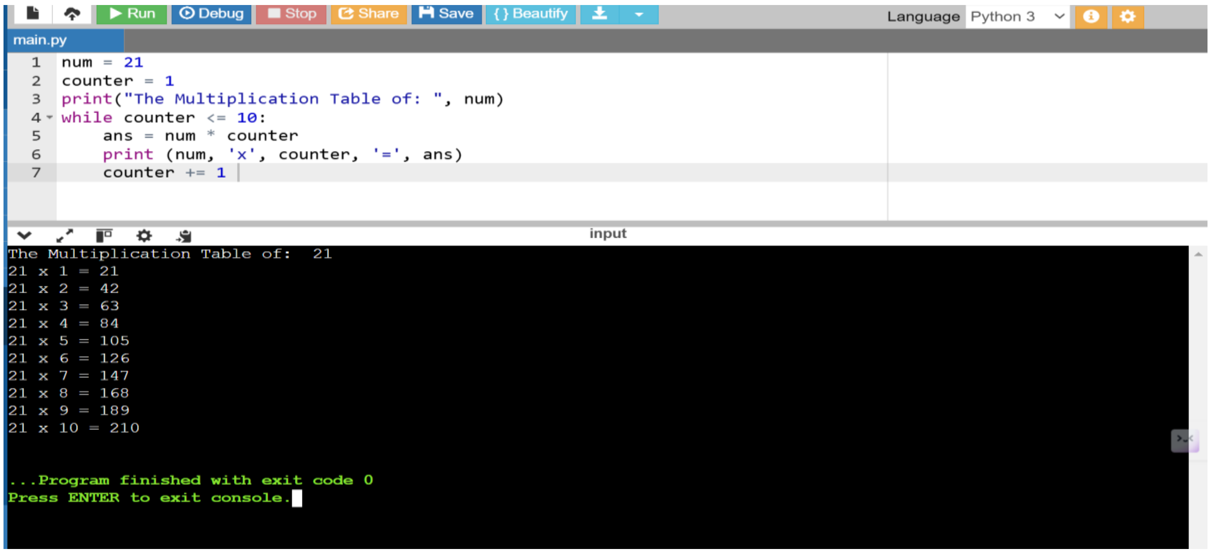
3. Armstrong number



```python
n = int(input())
n1=str(n)
l=len(n1)
temp=n
s=0
while n!=0:
    r=n%10
    s=s+(r**1)
    n=n//10
if s==temp:
    print("It is an Armstrong number")
else:
    print("It is not an Armstrong number ")
```

```
121
It is not an Armstrong number
```

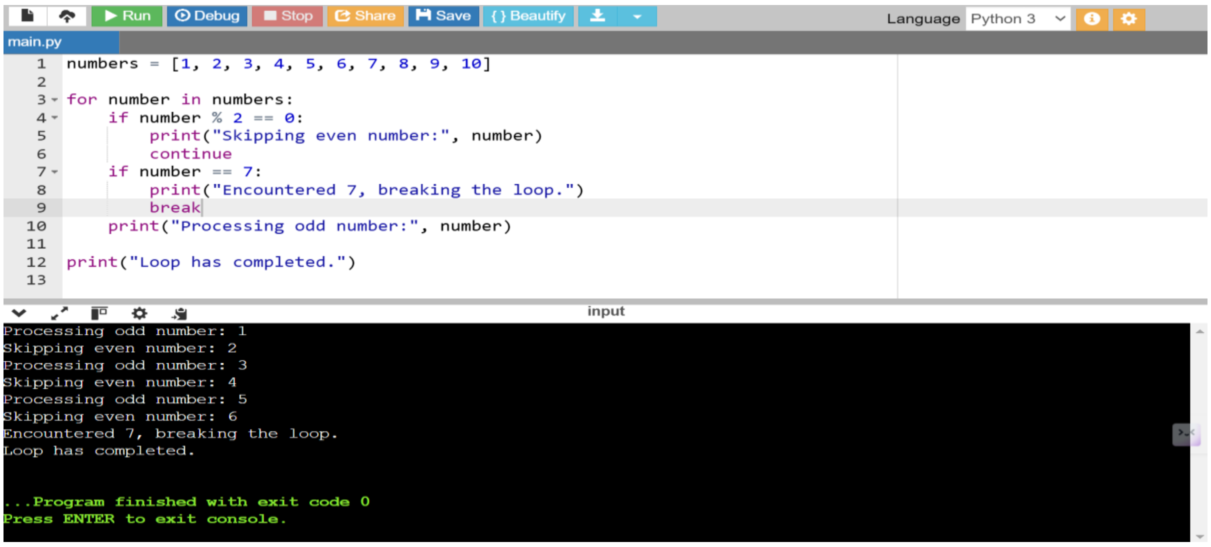
4. Multiplication Table



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

```
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


...Program finished with exit code 0
Press ENTER to exit console.
```

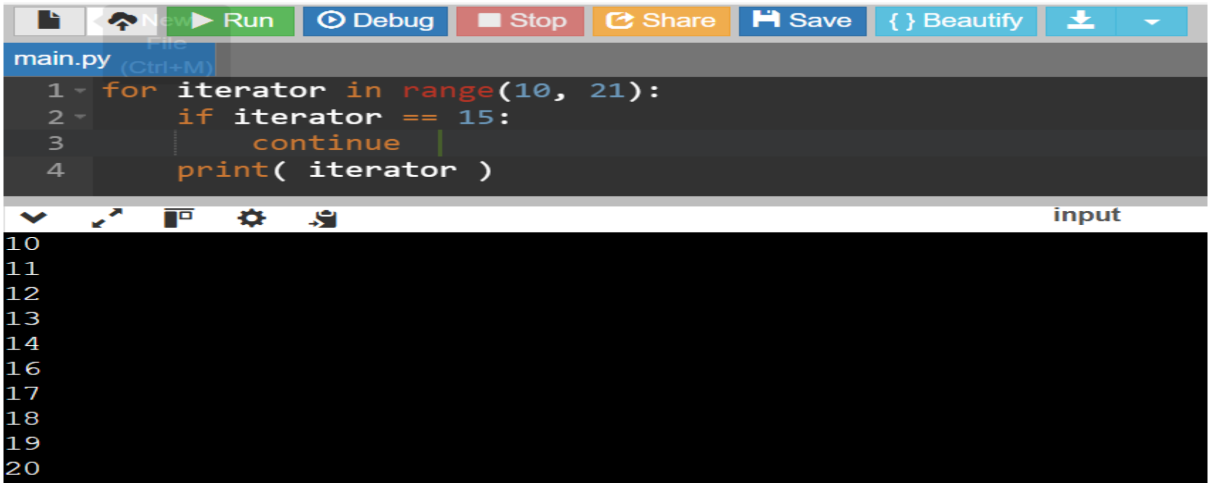
BREAK Statement:



```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for number in numbers:
    if number % 2 == 0:
        print("Skipping even number:", number)
        continue
    if number == 7:
        print("Encountered 7, breaking the loop.")
        break
    print("Processing odd number:", number)

print("Loop has completed.")
```

```
Processing odd number: 1
Skipping even number: 2
Processing odd number: 3
Skipping even number: 4
Processing odd number: 5
Skipping even number: 6
Encountered 7, breaking the loop.
Loop has completed.


...Program finished with exit code 0
Press ENTER to exit console.
```

CONTINUE Statement:

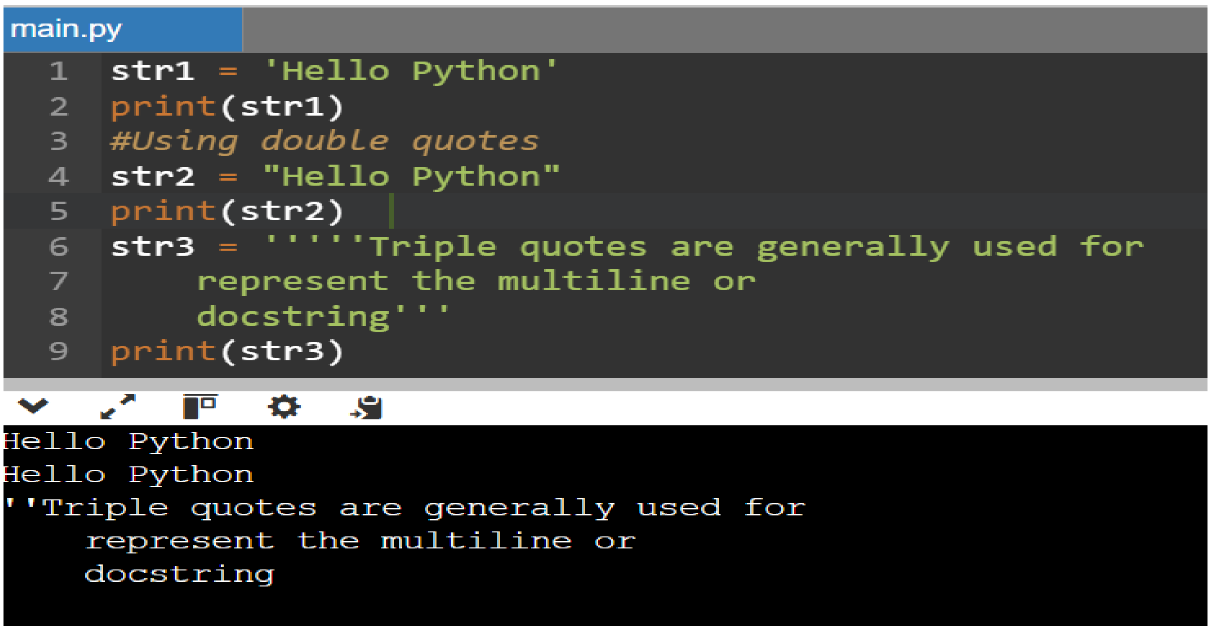


```python
for iterator in range(10, 21):
    if iterator == 15:
        continue
    print( iterator )
```

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

STRINGS:

1. **Creating a String in Python**

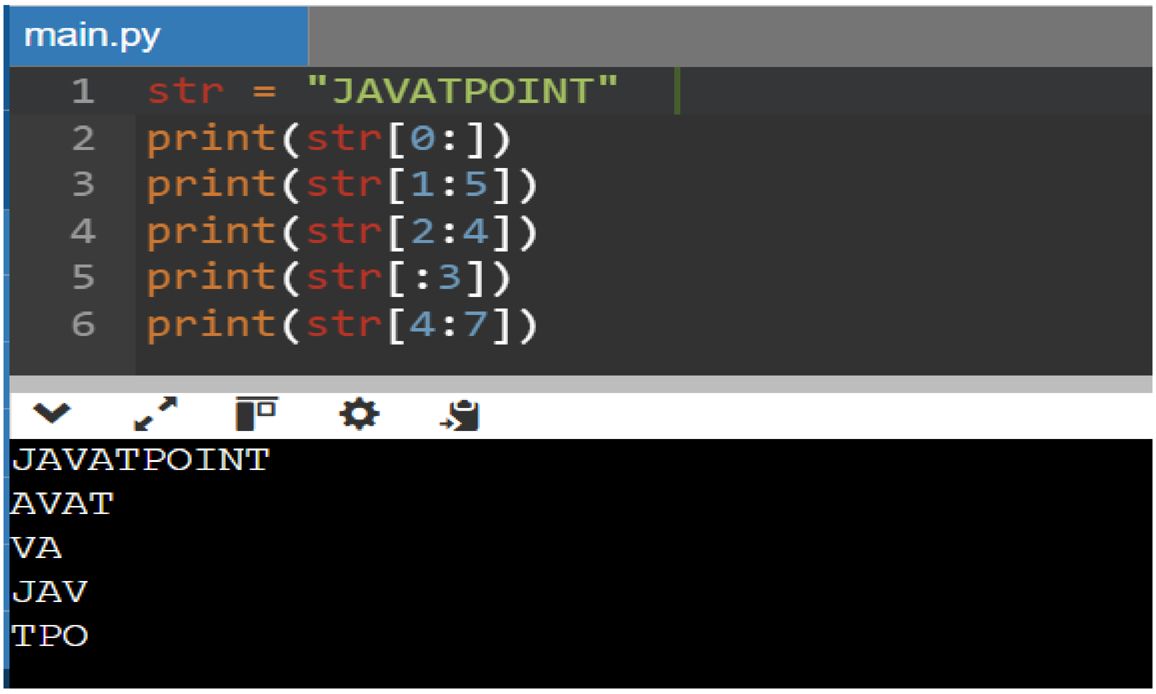


main.py

```python
str1 = 'Hello Python'
print(str1)
#Using double quotes
str2 = "Hello Python"
print(str2)
str3 = ''''Triple quotes are generally used for
    represent the multiline or
    docstring'''
print(str3)
```

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

2. **String Indexing**

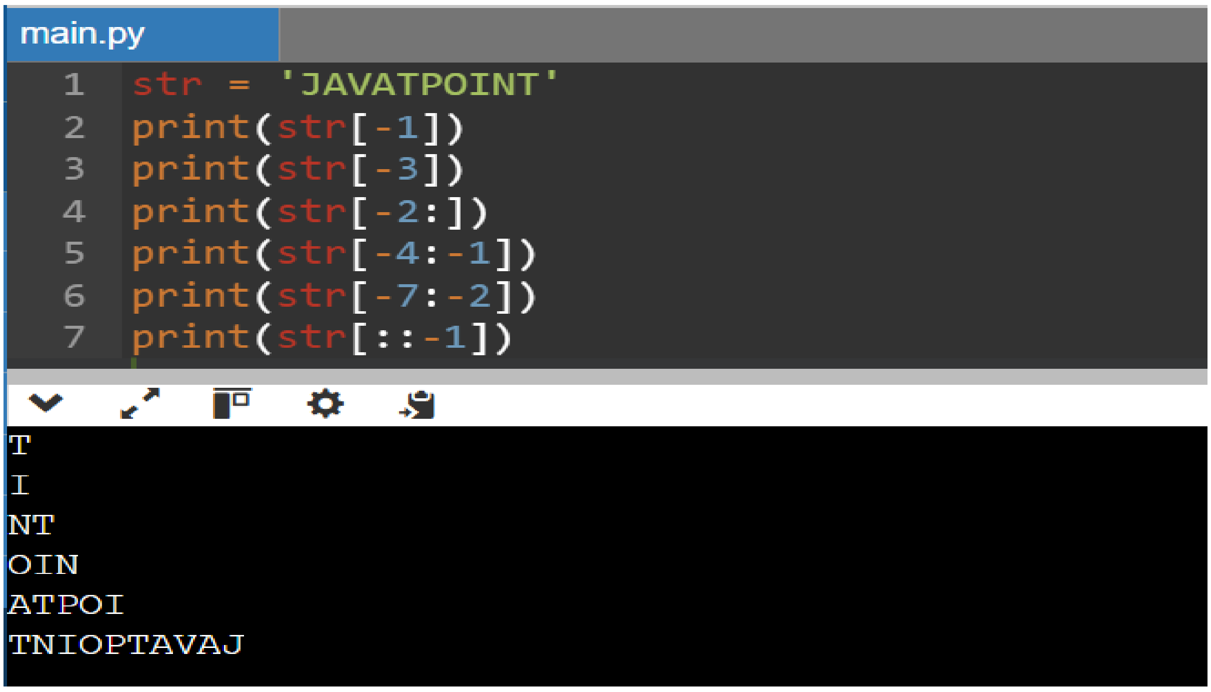


main.py

```python
str = "JAVATPOINT"
print(str[0:])
print(str[1:5])
print(str[2:4])
print(str[:3])
print(str[4:7])
```

```
JAVATPOINT
AVAT
VA
JAV
TPO
```

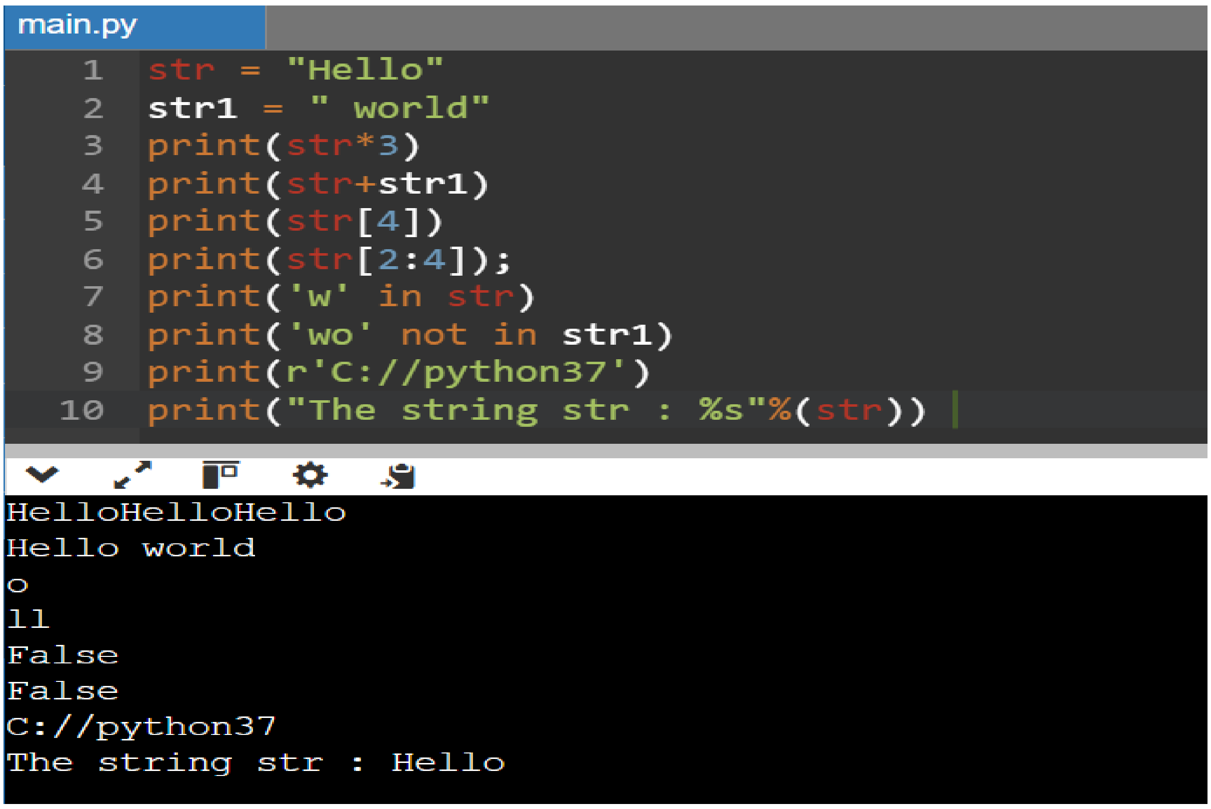
3. **String Splitting**



```
str = 'JAVATPOINT'
print(str[-1])
print(str[-3])
print(str[-2:])
print(str[-4:-1])
print(str[-7:-2])
print(str[::-1])
```

```
T
I
NT
OIN
ATPOI
TNIOPTAVAJ
```

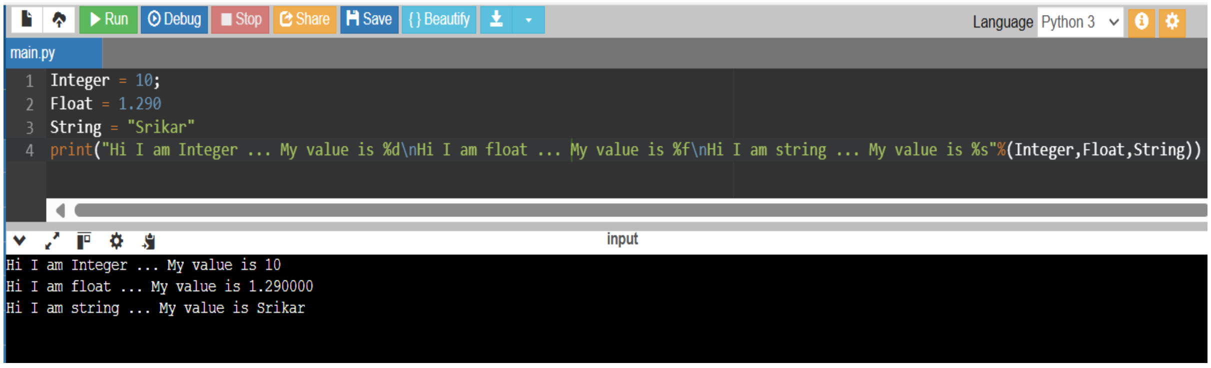
4. **Python String operators**



```
str = "Hello"
str1 = " world"
print(str*3)
print(str+str1)
print(str[4])
print(str[2:4]);
print('w' in str)
print('wo' not in str1)
print(r'C://python37')
print("The string str : %s"%(str))
```

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

5. Python string formatting using % operator

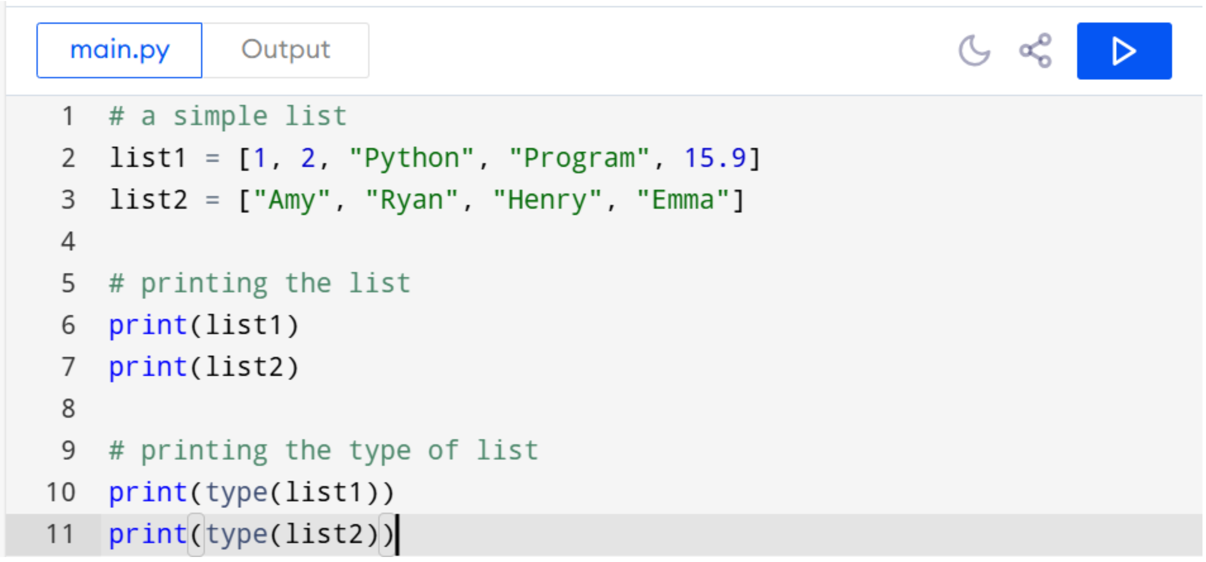


```python
Integer = 10;
Float = 1.290
String = "Srikar"
print("Hi I am Integer ... My value is %d\nHi I am float ... My value is %f\nHi I am string ... My value is %s"%(Integer,Float,String))
```

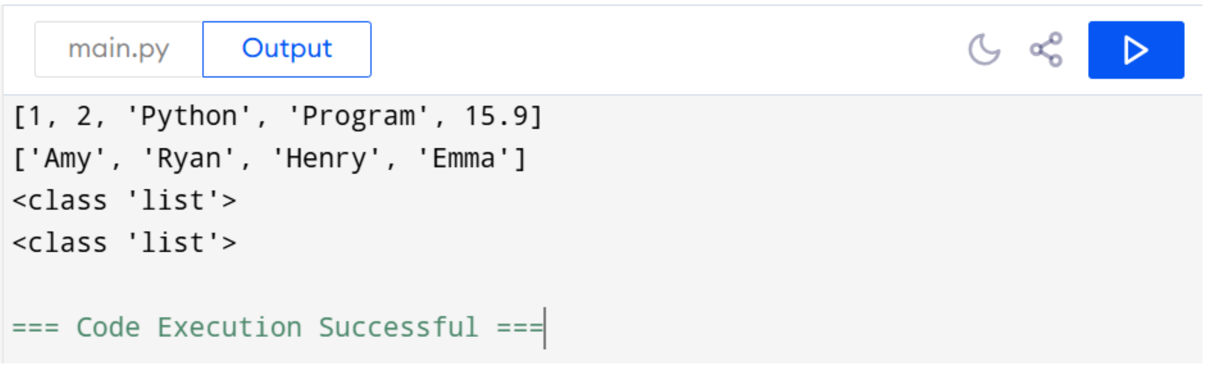
```
Hi I am Integer ... My value is 10
Hi I am float ... My value is 1.290000
Hi I am string ... My value is Srikar
```

PYTHON LISTS AND TUPLES

1. List Declaration



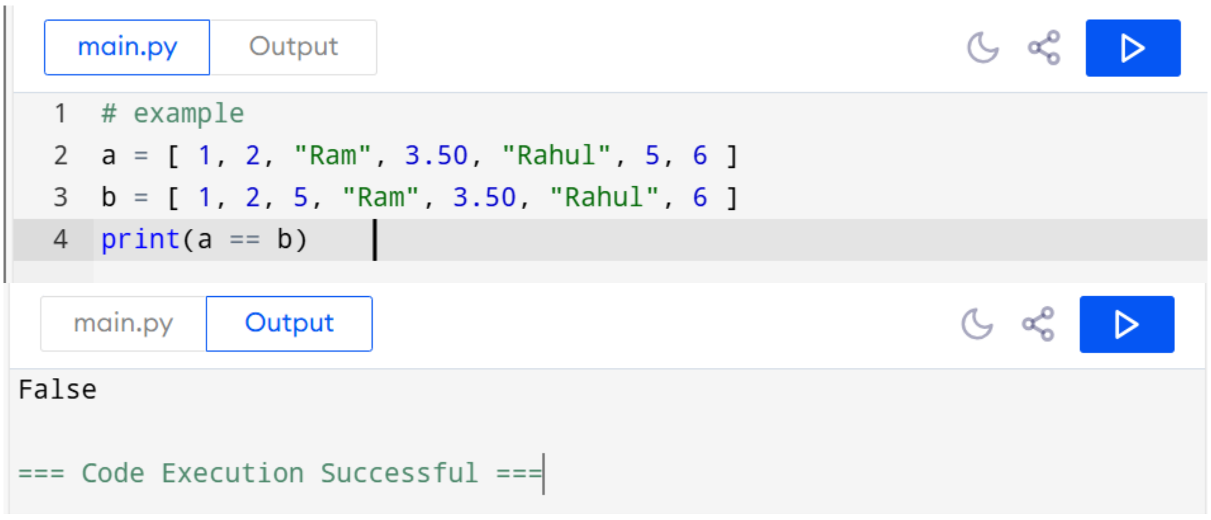
```python
# a simple list
list1 = [1, 2, "Python", "Program", 15.9]
list2 = ["Amy", "Ryan", "Henry", "Emma"]

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

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



```
[1, 2, 'Python', 'Program', 15.9]
['Amy', 'Ryan', 'Henry', 'Emma']
<class 'list'>
<class 'list'>

=== Code Execution Successful ===
```

2. Ordered List Checking

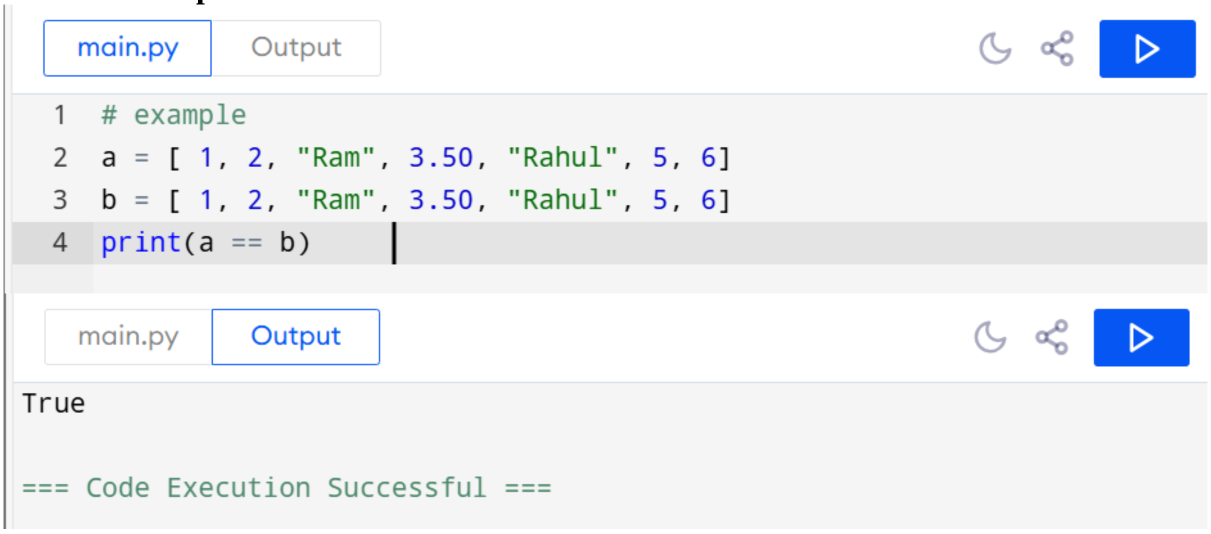
Example 1:



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

```
False

=== Code Execution Successful ===
```

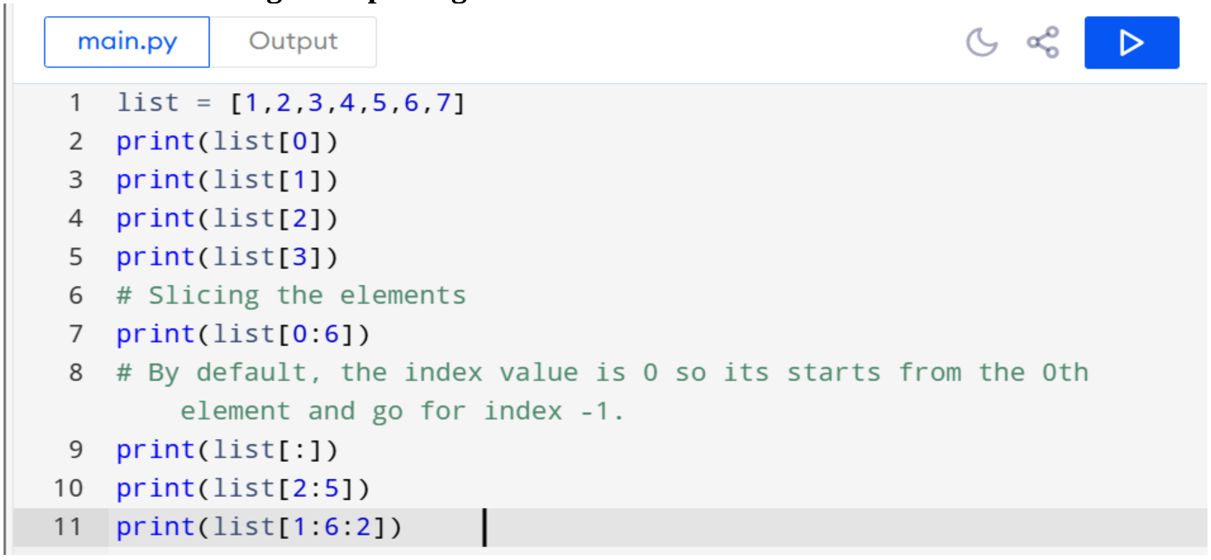
Example 2:



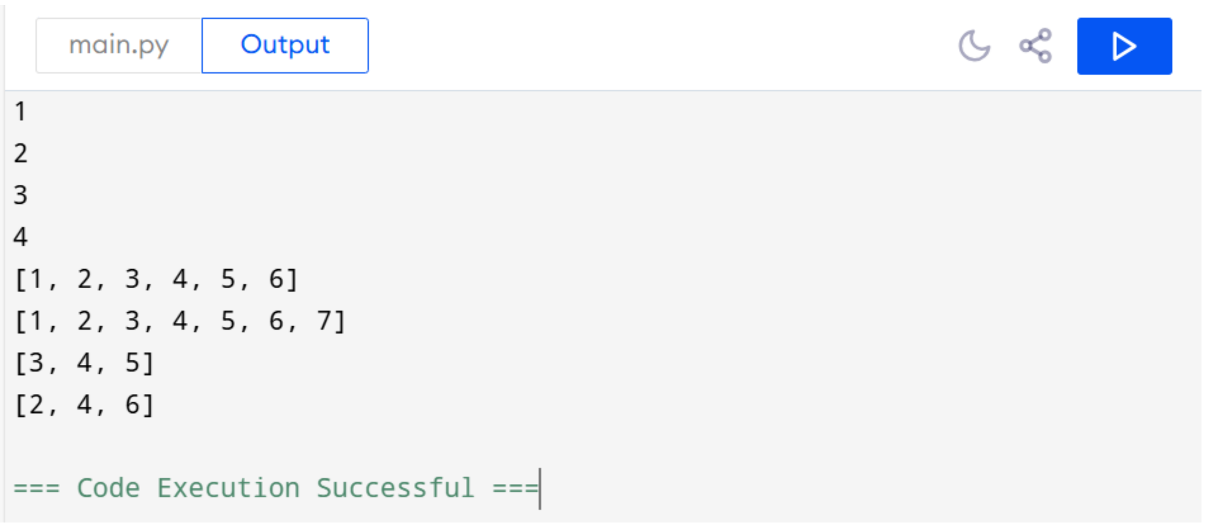
```
# example
a = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6]
b = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6]
print(a == b)
```

```
True

=== Code Execution Successful ===
```

3. List Indexing and Splitting

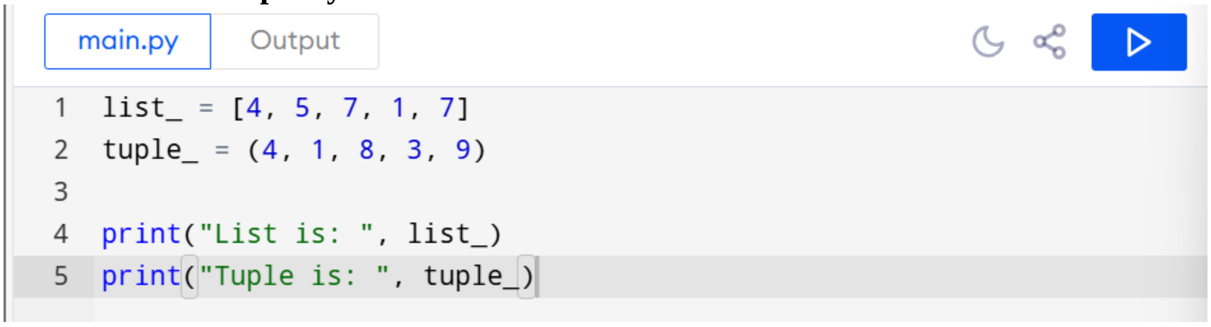


```
list = [1,2,3,4,5,6,7]
print(list[0])
print(list[1])
print(list[2])
print(list[3])
# Slicing the elements
print(list[0:6])
# By default, the index value is 0 so its starts from the 0th
    element and go for index -1.
print(list[:])
print(list[2:5])
print(list[1:6:2])
```

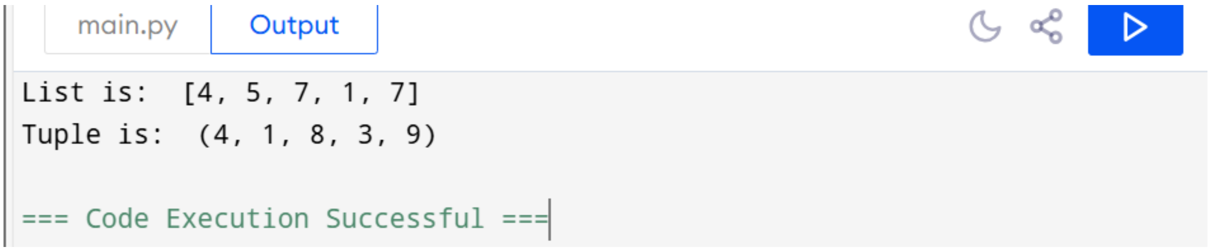


```
1
2
3
4
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6, 7]
[3, 4, 5]
[2, 4, 6]

=== Code Execution Successful ===
```

4. **List and Tuple Syntax Difference**

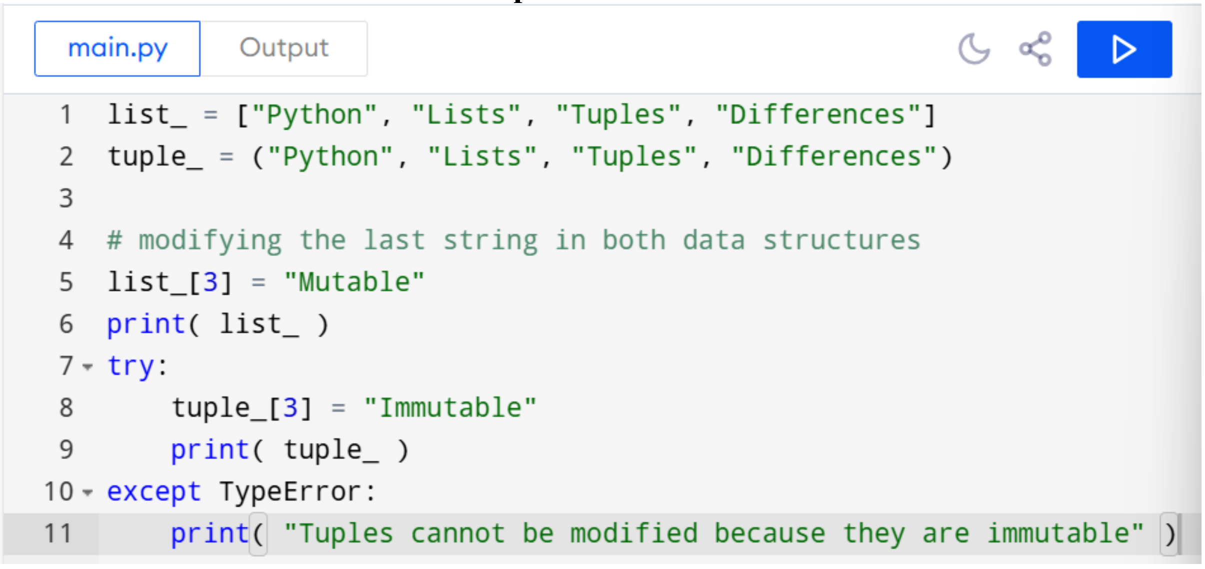


```python
list_ = [4, 5, 7, 1, 7]
tuple_ = (4, 1, 8, 3, 9)

print("List is: ", list_)
print("Tuple is: ", tuple_)
```

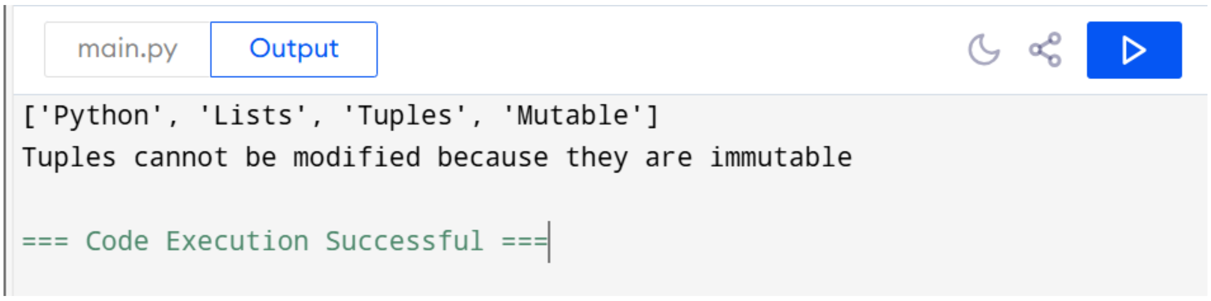


```
List is:  [4, 5, 7, 1, 7]
Tuple is:  (4, 1, 8, 3, 9)

=== Code Execution Successful ===
```

5. **Mutable List vs Immutable Tuple**

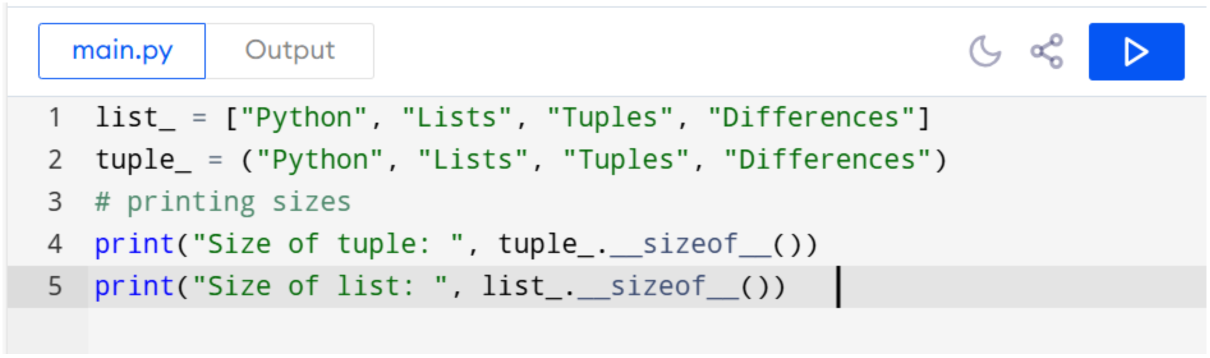


```python
list_ = ["Python", "Lists", "Tuples", "Differences"]
tuple_ = ("Python", "Lists", "Tuples", "Differences")

# modifying the last string in both data structures
list_[3] = "Mutable"
print( list_ )
try:
    tuple_[3] = "Immutable"
    print( tuple_ )
except TypeError:
    print( "Tuples cannot be modified because they are immutable" )
```

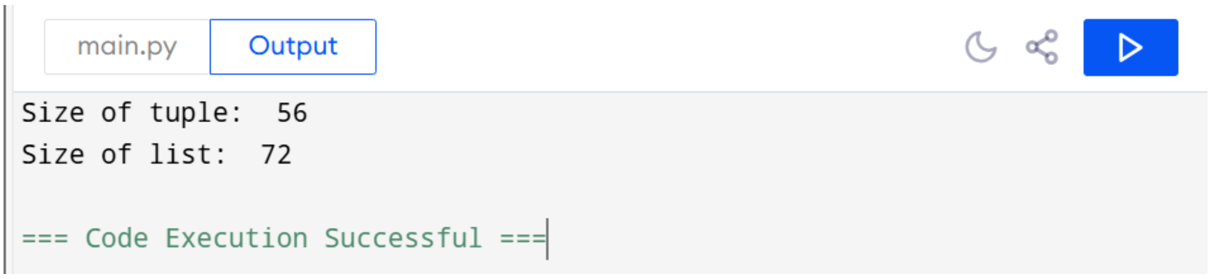


```
['Python', 'Lists', 'Tuples', 'Mutable']
Tuples cannot be modified because they are immutable

=== Code Execution Successful ===
```

6. **Size Difference**



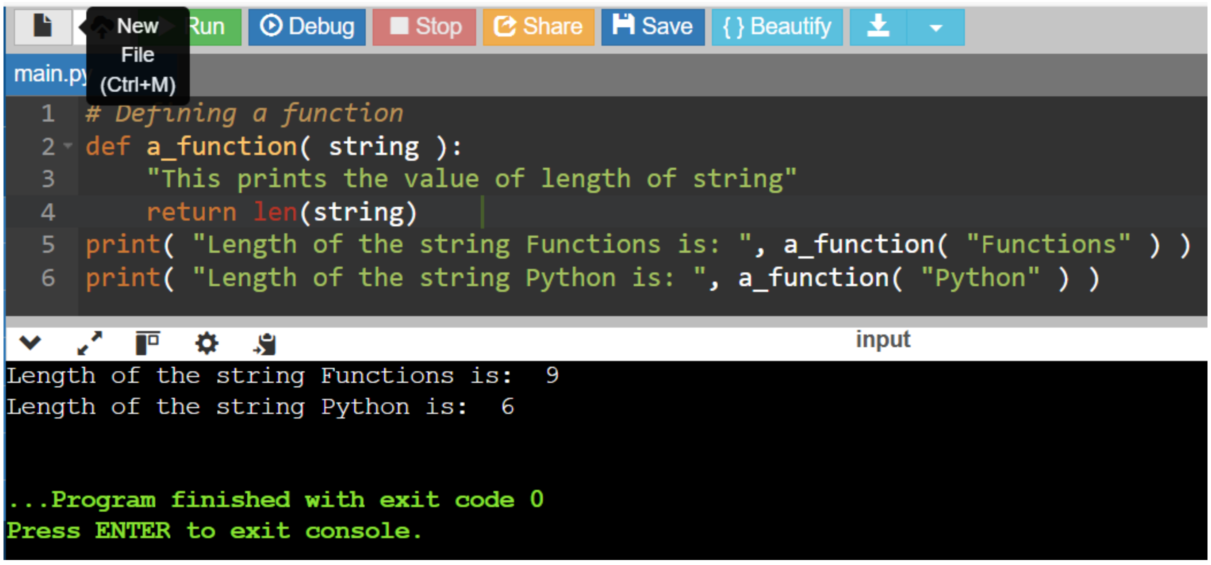
```python
list_ = ["Python", "Lists", "Tuples", "Differences"]
tuple_ = ("Python", "Lists", "Tuples", "Differences")
# printing sizes
print("Size of tuple: ", tuple_.__sizeof__())
print("Size of list: ", list_.__sizeof__())
```



```
Size of tuple:  56
Size of list:  72

=== Code Execution Successful ===
```

PYTHON FUNCTIONS

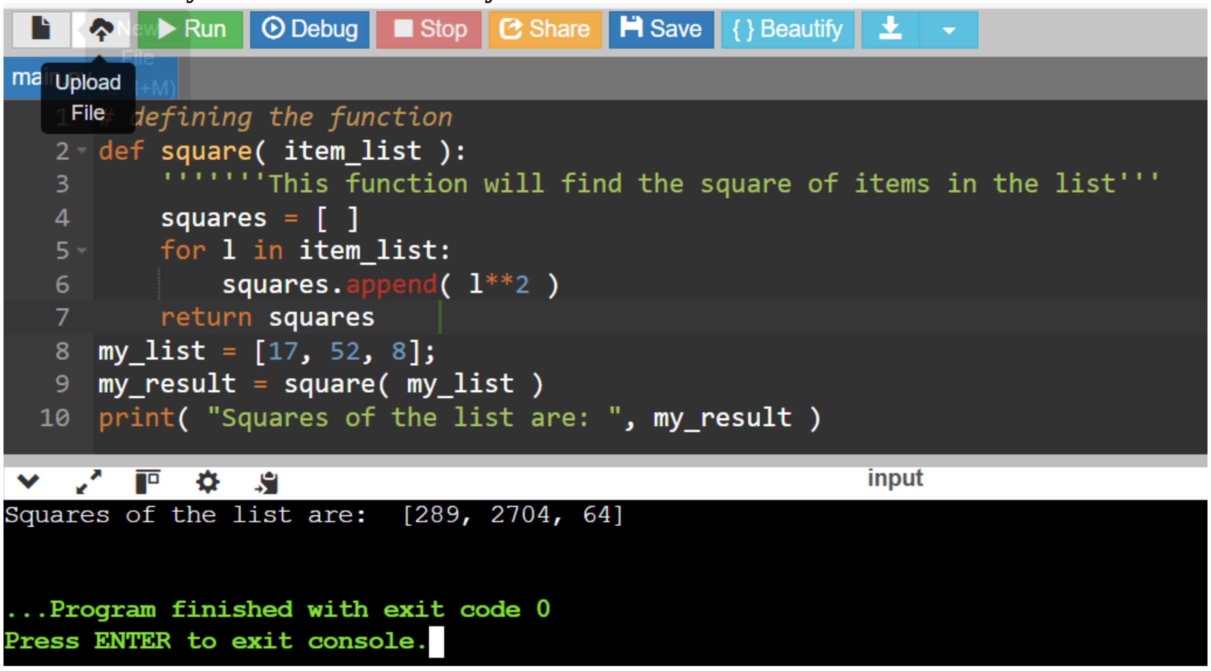
1. **Calling a function**



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


...Program finished with exit code 0
Press ENTER to exit console.
```

2. **Pass by Reference Vs Pass by Value**

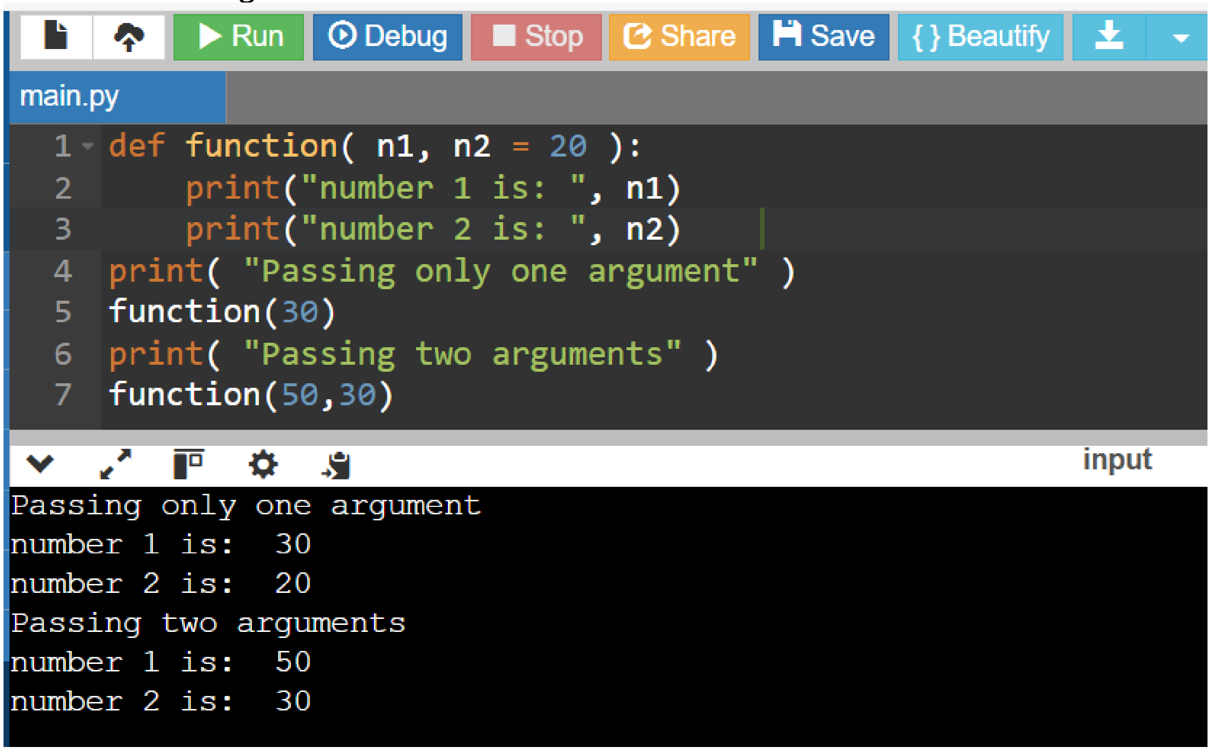


```
# defining the function
def square( item_list ):
    ''''''This function will find the square of items in the list'''
    squares = [ ]
    for l in item_list:
        squares.append( l**2 )
    return squares
my_list = [17, 52, 8];
my_result = square( my_list )
print( "Squares of the list are: ", my_result )
```

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


...Program finished with exit code 0
Press ENTER to exit console.
```

FUNCTION ARGUMENTS

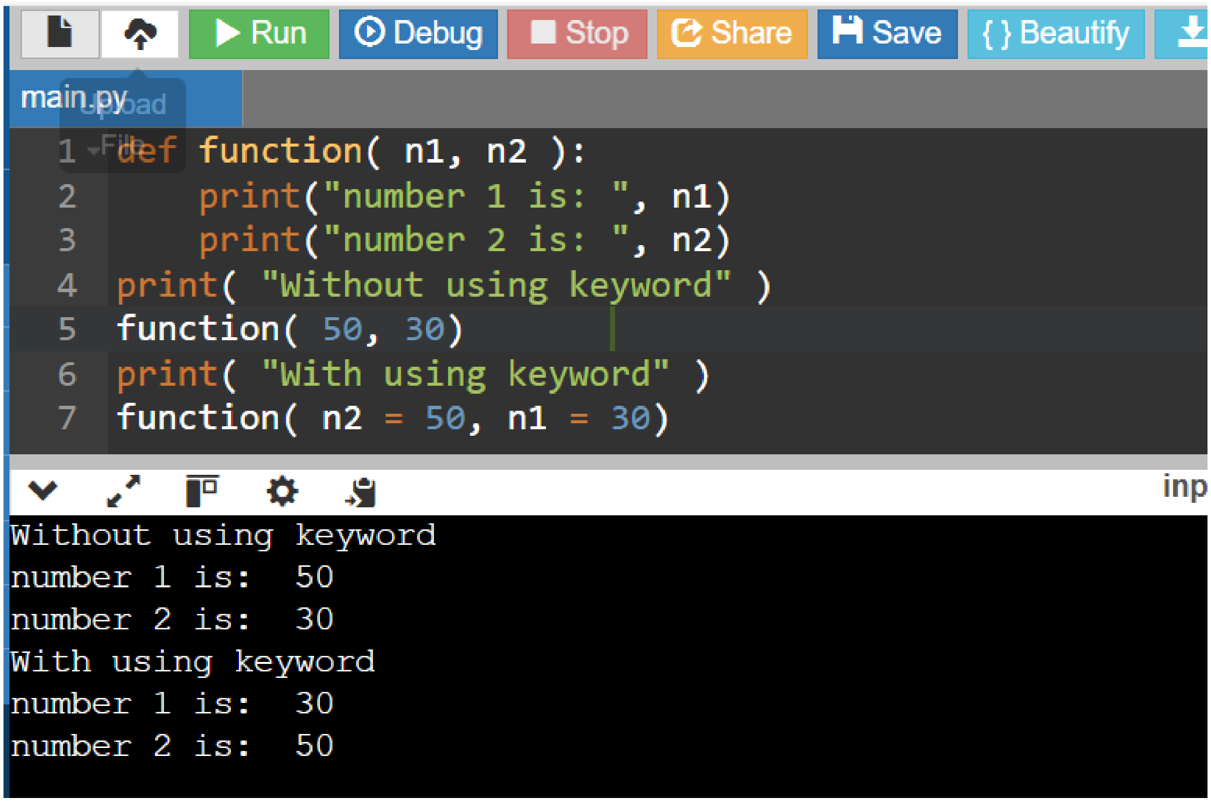
1. **Default arguments**



```
def function( n1, n2 = 20 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)
print( "Passing only one argument" )
function(30)
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
```

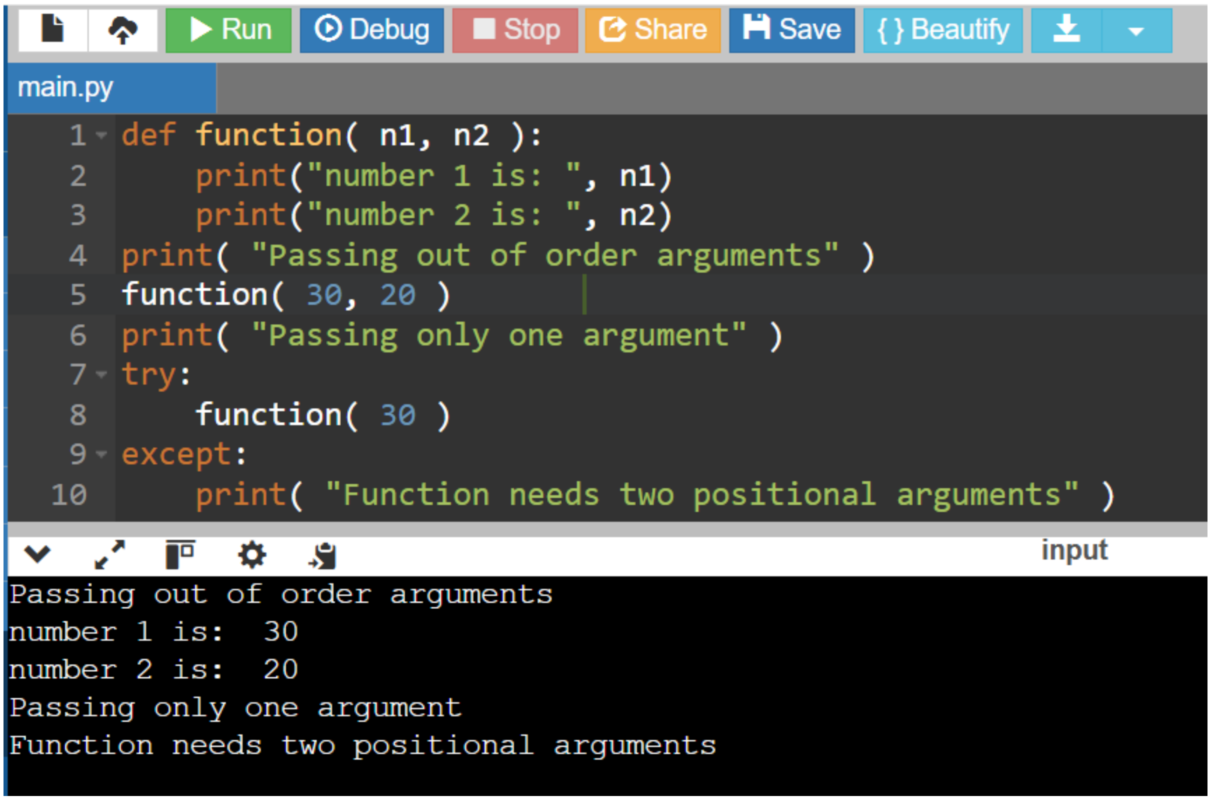
2. **Keyword arguments**



```python
def function( n1, n2 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)
print( "Without using keyword" )
function( 50, 30)
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
```

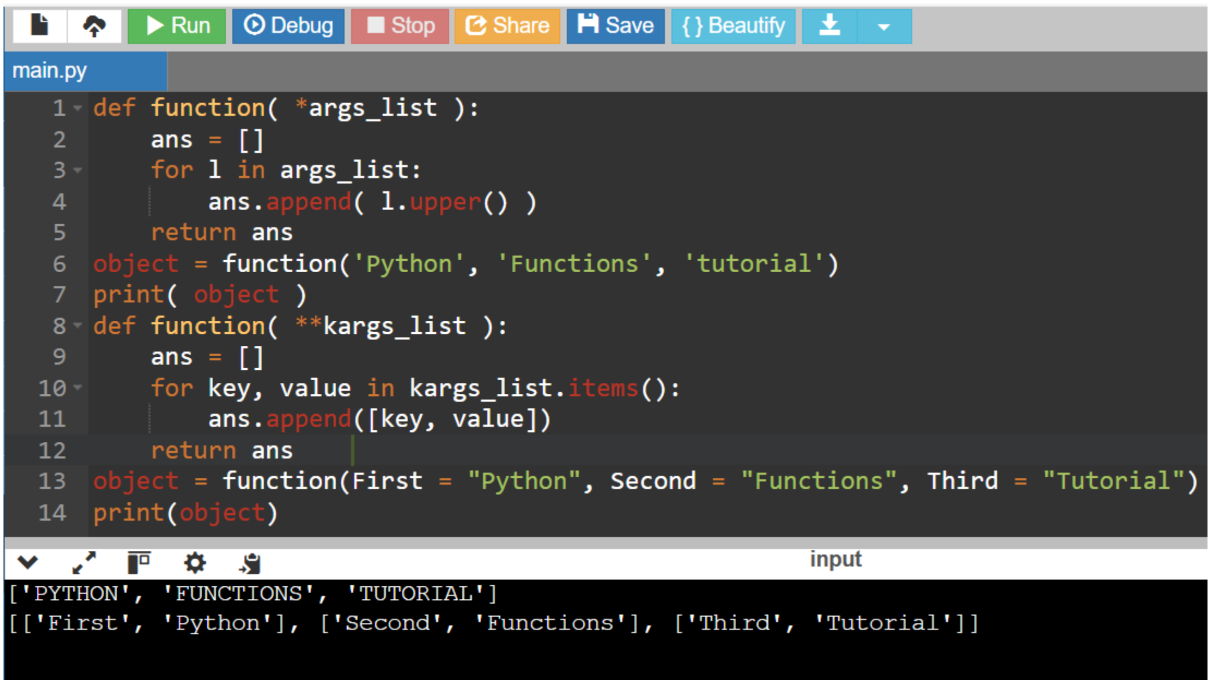
3. **Required arguments**



```python
def function( n1, n2 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)
print( "Passing out of order arguments" )
function( 30, 20 )
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
```

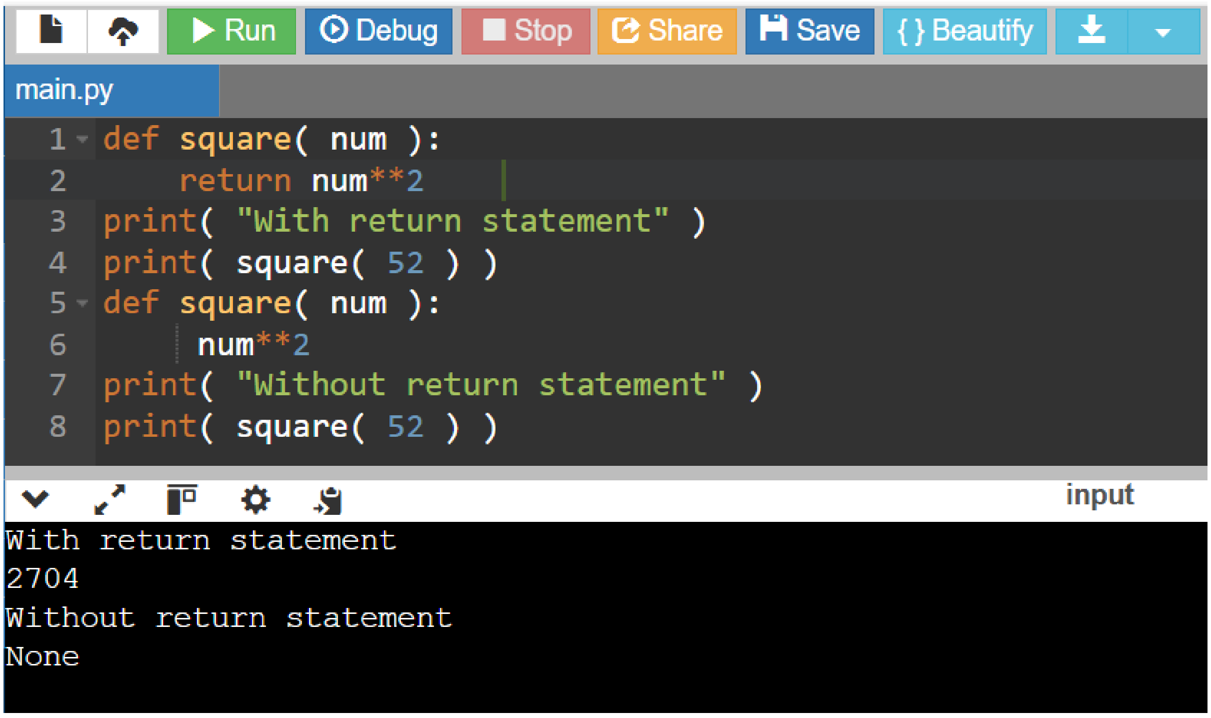
4. **Variable-length arguments**



```python
def function( *args_list ):
    ans = []
    for l in args_list:
        ans.append( l.upper() )
    return ans
object = function('Python', 'Functions', 'tutorial')
print( object )
def function( **kargs_list ):
    ans = []
    for key, value in kargs_list.items():
        ans.append([key, value])
    return ans
object = function(First = "Python", Second = "Functions", Third = "Tutorial")
print(object)
```

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

RETURN STATEMENT

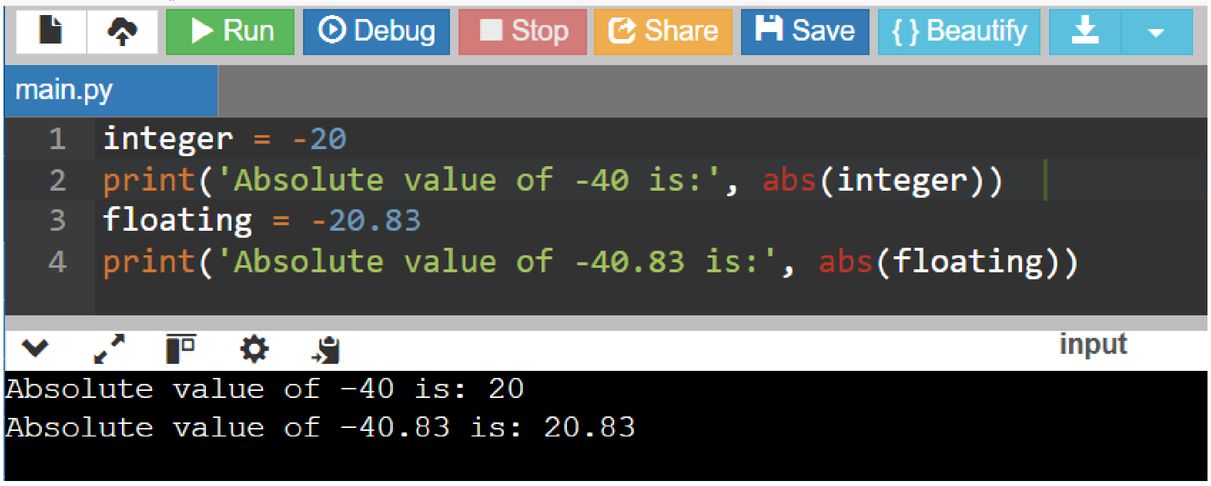


```python
def square( num ):
    return num**2
print( "With return statement" )
print( square( 52 ) )
def square( num ):
    num**2
print( "Without return statement" )
print( square( 52 ) )
```

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

PYTHON BUILT-IN FUNCTIONS

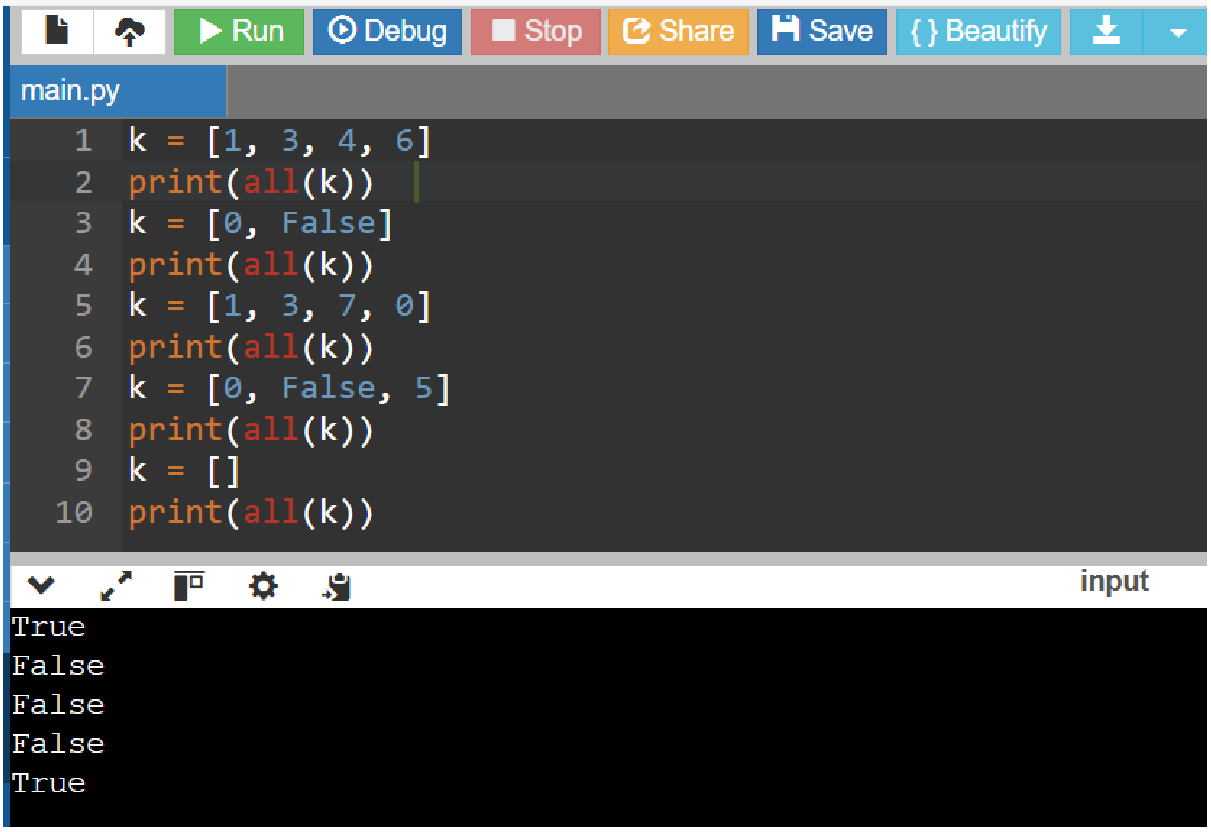
1. **Abs () function**



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

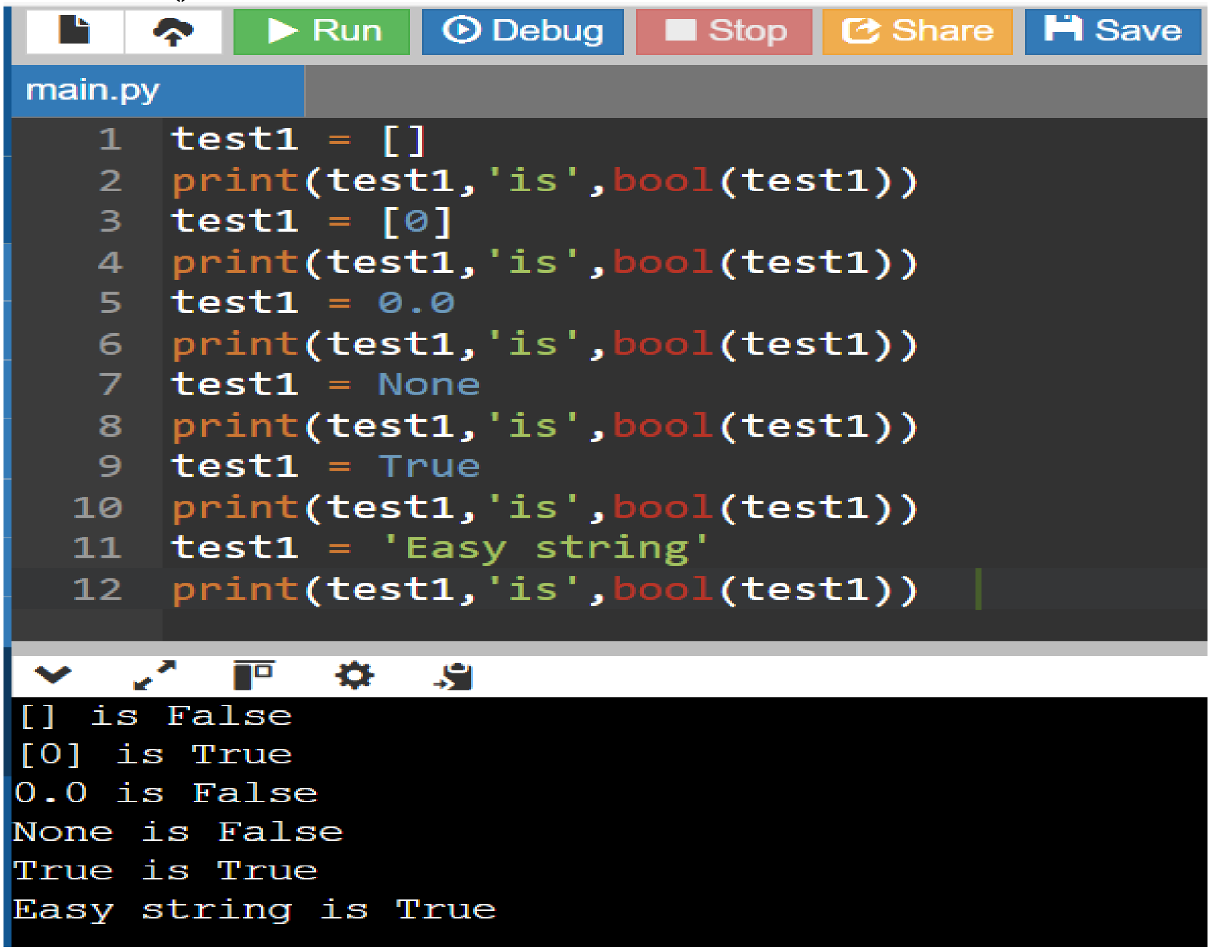
2. **All () function**



```
k = [1, 3, 4, 6]
print(all(k))
k = [0, False]
print(all(k))
k = [1, 3, 7, 0]
print(all(k))
k = [0, False, 5]
print(all(k))
k = []
print(all(k))
```

```
True
False
False
False
True
```

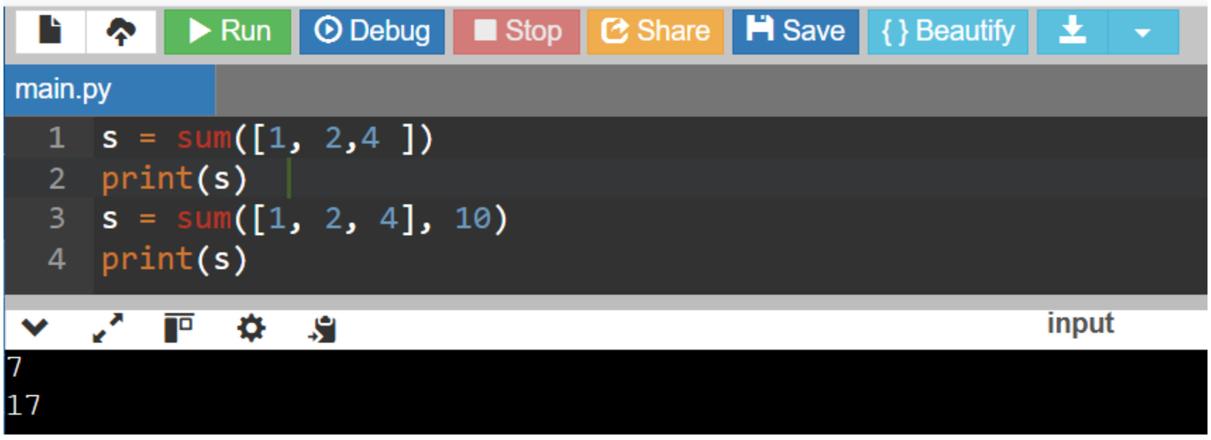
3. **Bool () function**



```
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
```

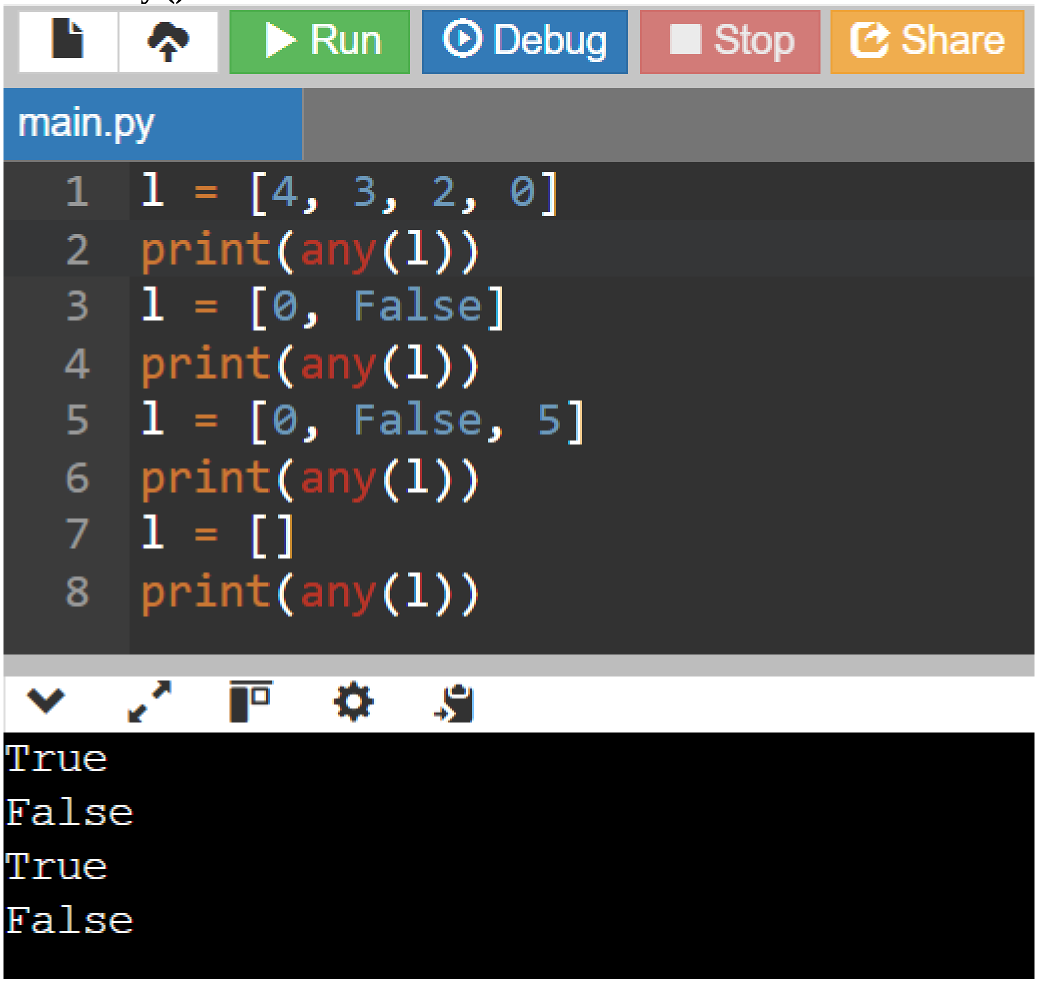
4. **Sum () Function**



```
s = sum([1, 2,4 ])
print(s)
s = sum([1, 2, 4], 10)
print(s)
```

```
7
17
```

5. **Any () function**

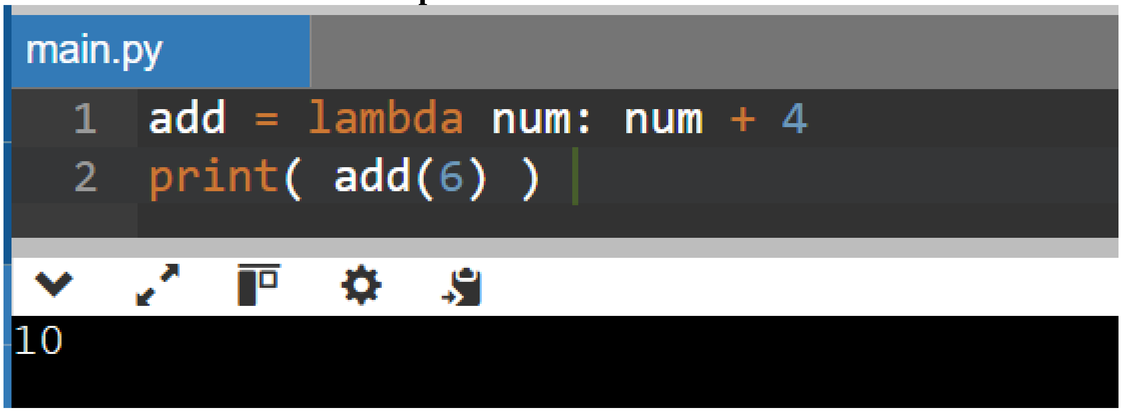


```
l = [4, 3, 2, 0]
print(any(l))
l = [0, False]
print(any(l))
l = [0, False, 5]
print(any(l))
l = []
print(any(l))
```

```
True
False
True
False
```

PYTHON LAMBDA FUNCTION

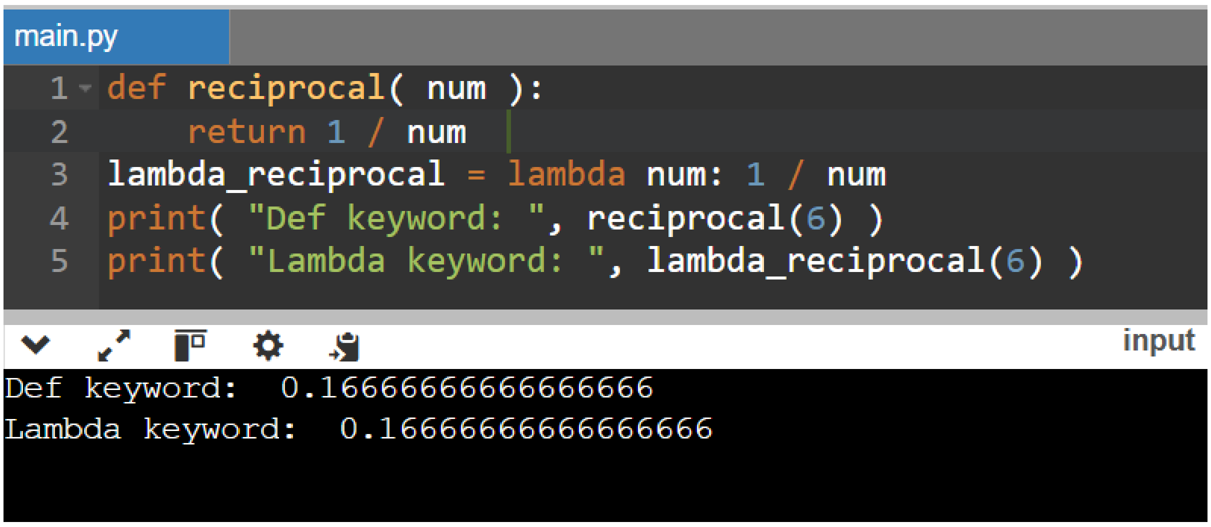
1. **Lambda function example**



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

```
10
```

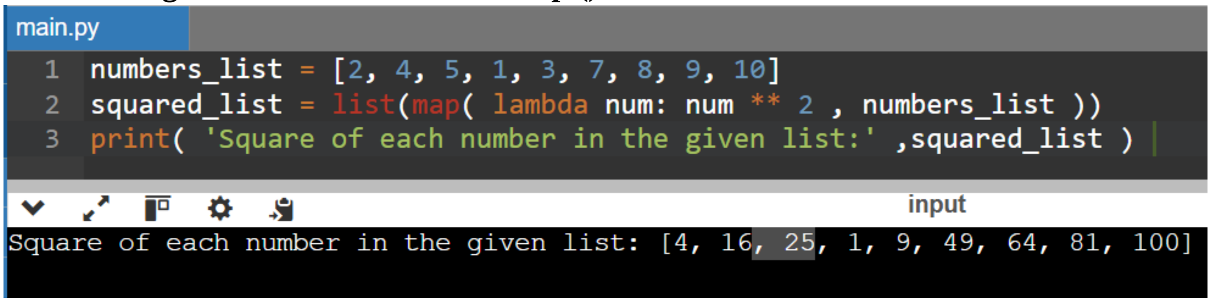
2. Distinction between Lambda and Def Function



```python
def reciprocal( num ):
    return 1 / num
lambda_reciprocal = lambda num: 1 / num
print( "Def keyword: ", reciprocal(6) )
print( "Lambda keyword: ", lambda_reciprocal(6) )
```

```
Def keyword:  0.16666666666666666
Lambda keyword:  0.16666666666666666
```

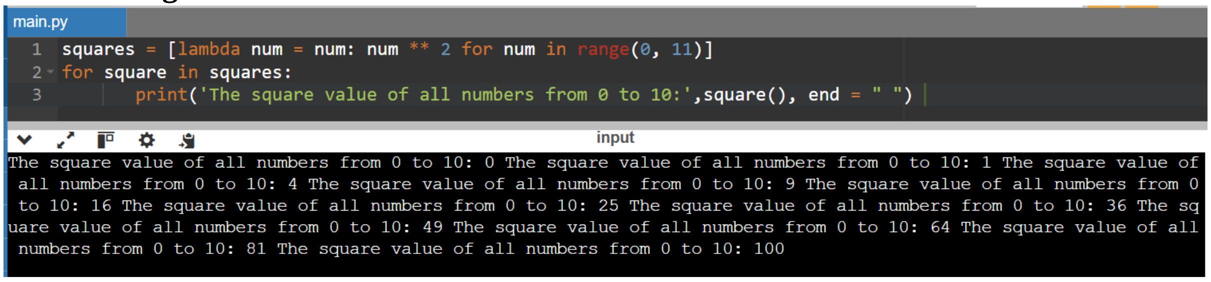
3. Using Lambda Function with map ()



```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]
```

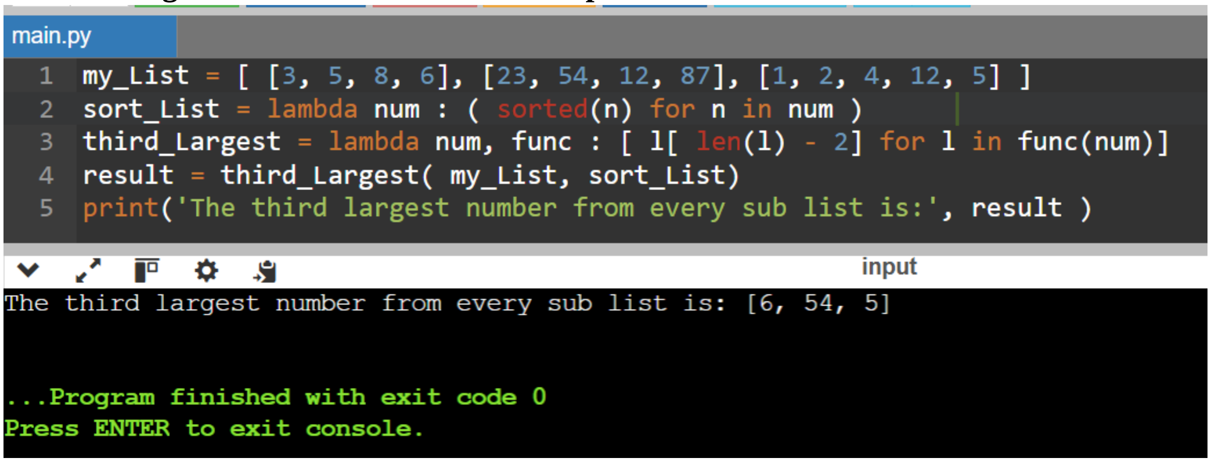
4. Using Lambda Function with List



```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 sq
uare 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 value of all numbers from 0 to 10: 100
```

5. Using Lambda Function with Multiple Statements



```python
my_List = [ [3, 5, 8, 6], [23, 54, 12, 87], [1, 2, 4, 12, 5] ]
sort_List = lambda num : ( sorted(n) for n in num )
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]


...Program finished with exit code 0
Press ENTER to exit console.
```