

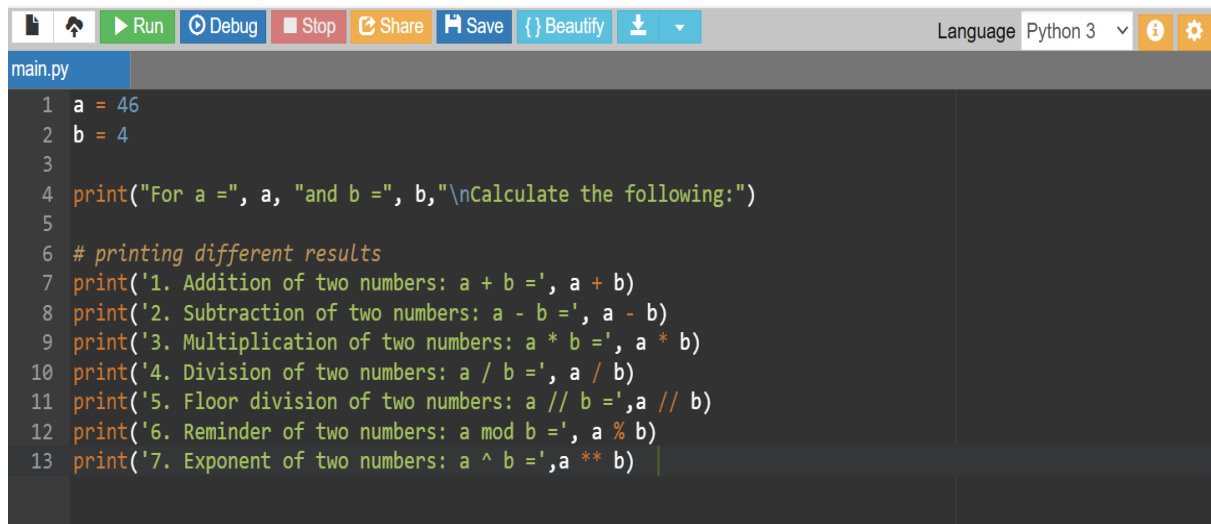
# PYTHON ASSIGNMENT

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

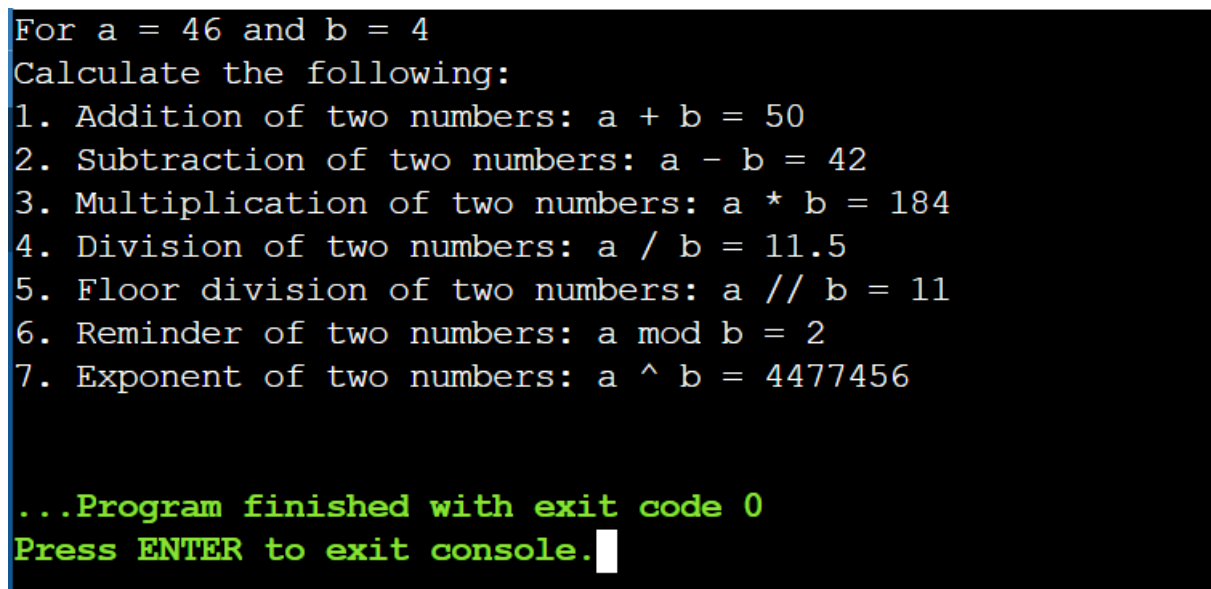
### 1. Arithmetic Operators

#### CODE

A screenshot of a Python IDE interface. The top toolbar contains icons for Run, Debug, Stop, Share, Save, Beautify, and a dropdown menu. The language is set to Python 3. The file is named main.py. The code is as follows:

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

#### OUTPUT

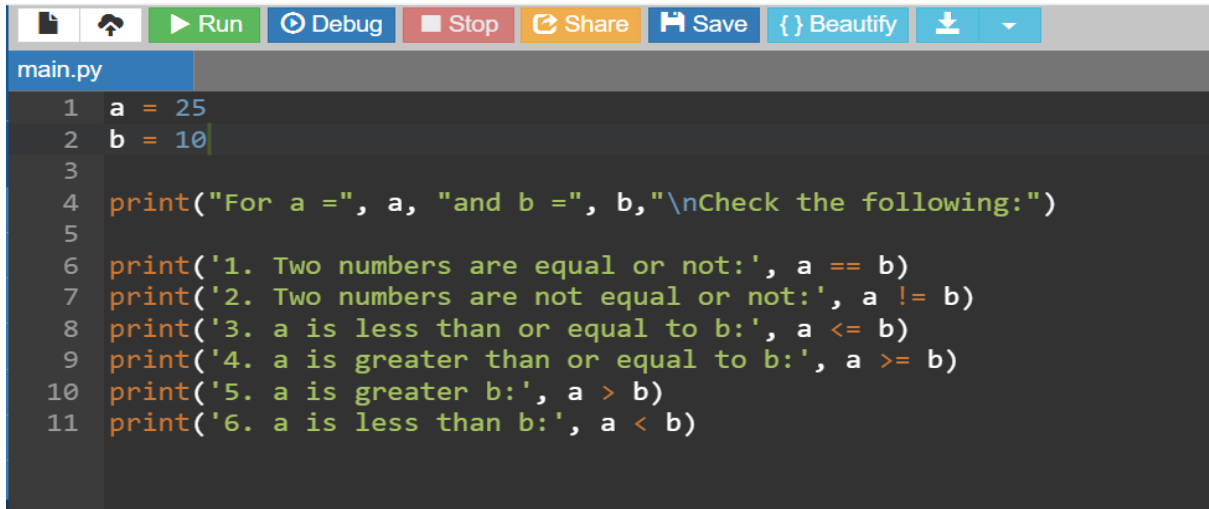
A screenshot of a terminal window showing the output of the Python program. The output is as follows:

```
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



```
main.py
1 a = 25
2 b = 10
3
4 print("For a =", a, "and b =", b, "\nCheck the following:")
5
6 print('1. Two numbers are equal or not:', a == b)
7 print('2. Two numbers are not equal or not:', a != b)
8 print('3. a is less than or equal to b:', a <= b)
9 print('4. a is greater than or equal to b:', a >= b)
10 print('5. a is greater b:', a > b)
11 print('6. a is less than b:', a < b)
```

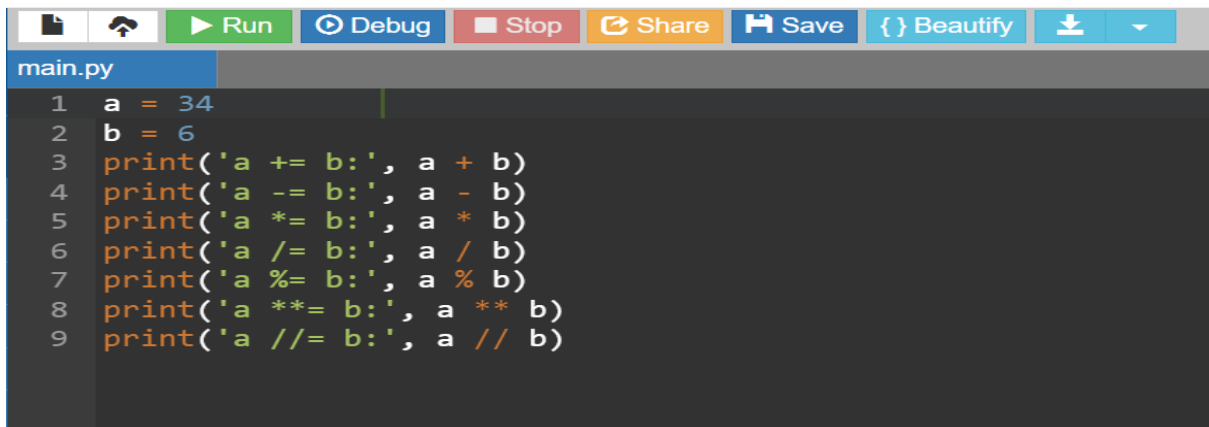
### OUTPUT

```
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.
```

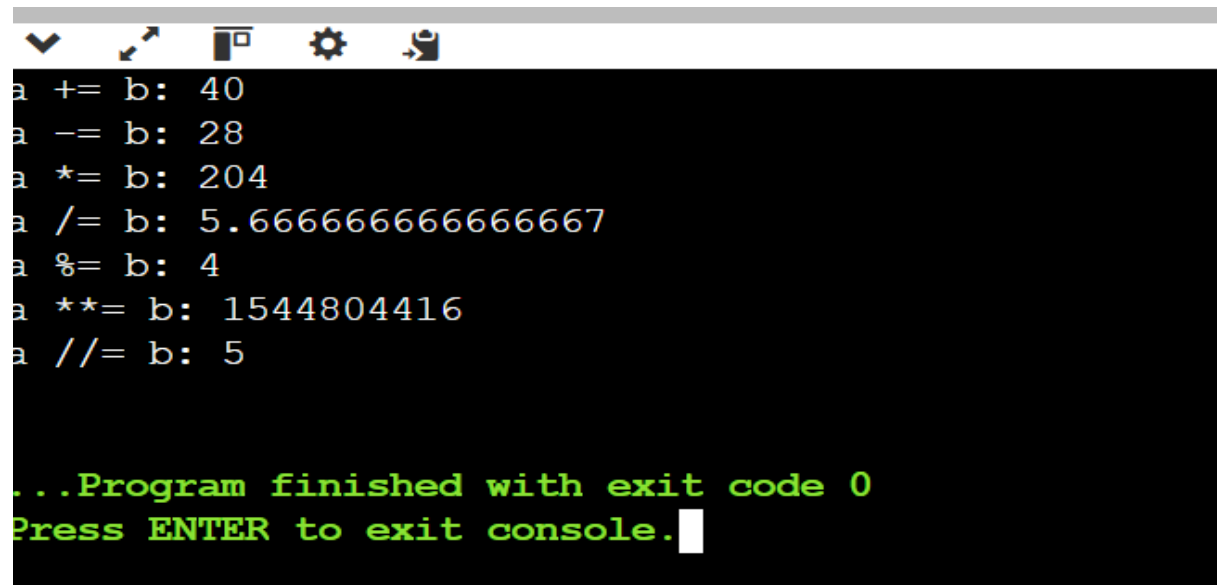
## 3. Assignment Operators

### CODE



```
main.py
1 a = 34
2 b = 6
3 print('a += b:', a + b)
4 print('a -= b:', a - b)
5 print('a *= b:', a * b)
6 print('a /= b:', a / b)
7 print('a %= b:', a % b)
8 print('a **= b:', a ** b)
9 print('a //= b:', a // b)
```

## OUTPUT



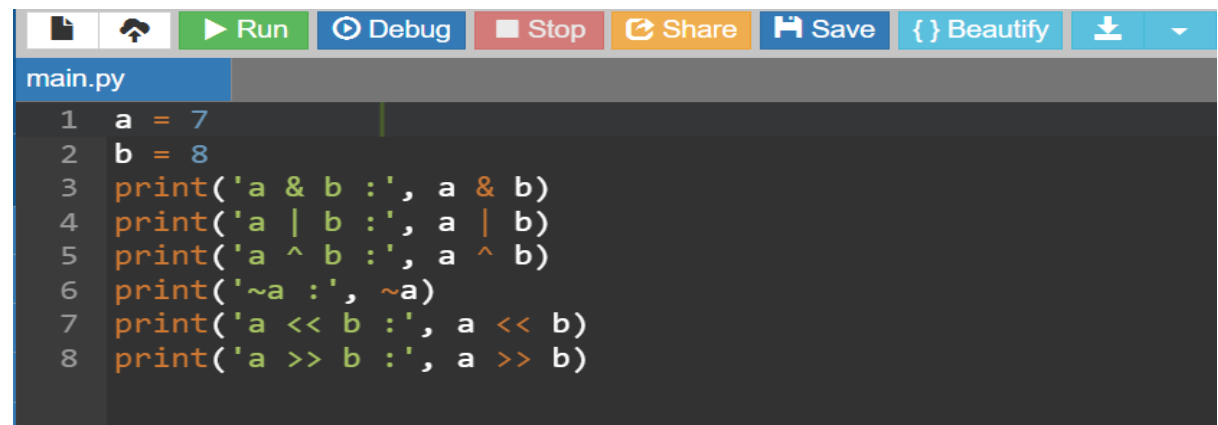
```

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

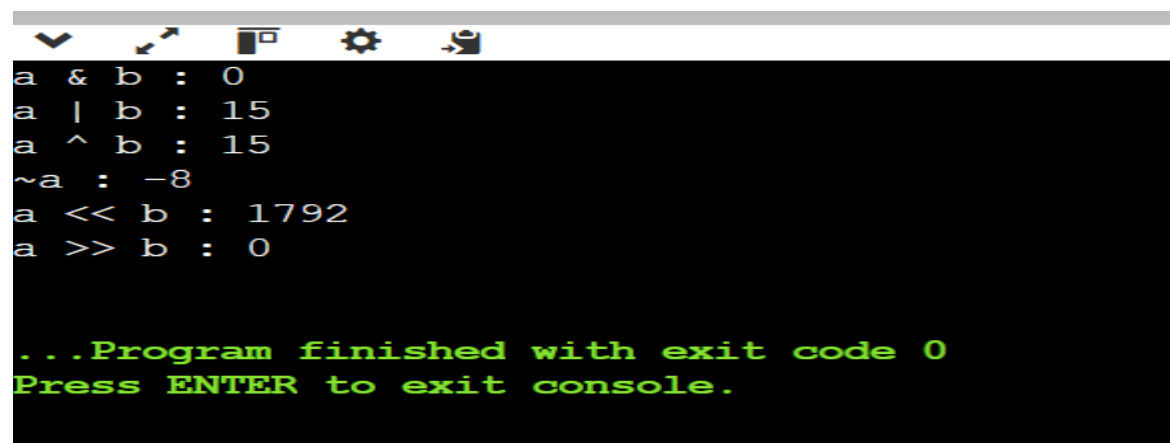
### CODE



```

main.py
1 a = 7
2 b = 8
3 print('a & b :', a & b)
4 print('a | b :', a | b)
5 print('a ^ b :', a ^ b)
6 print('~a :', ~a)
7 print('a << b :', a << b)
8 print('a >> b :', a >> b)
```

## OUTPUT



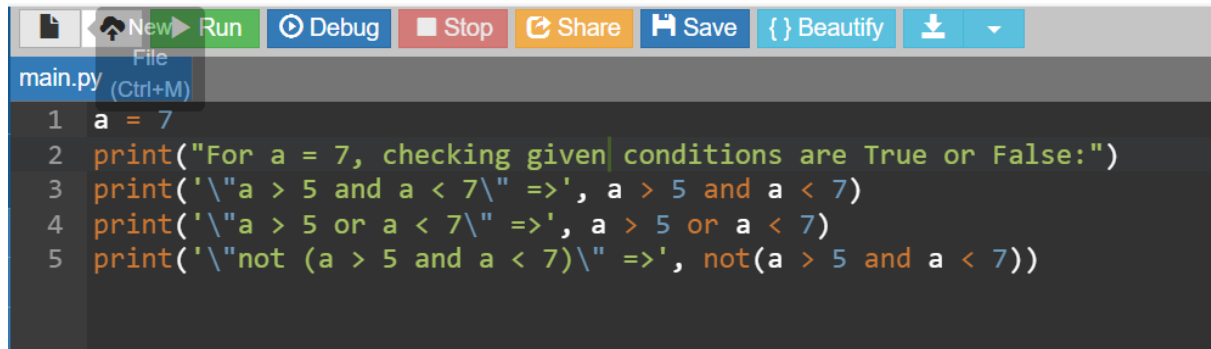
```

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.
```

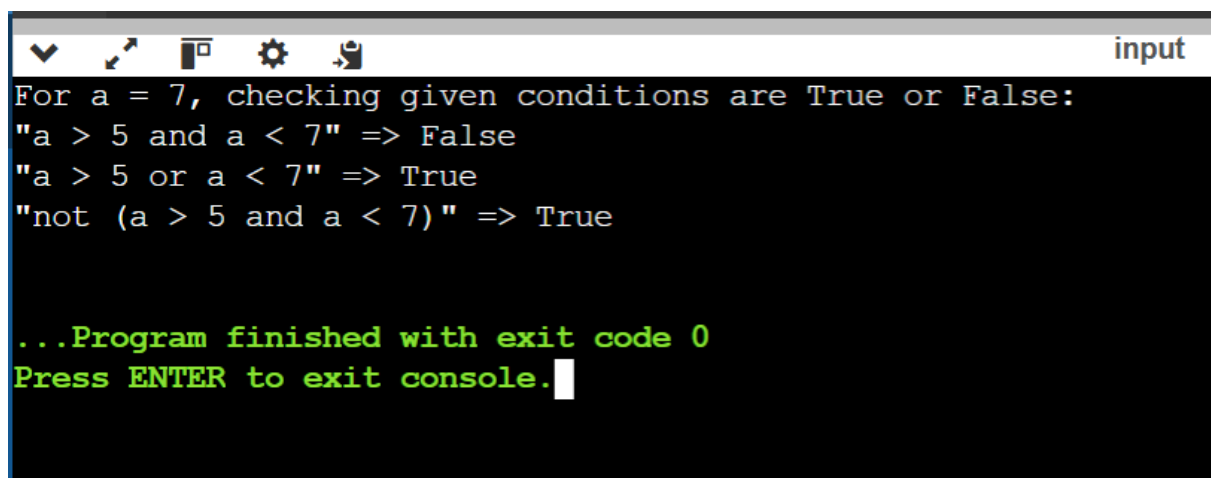
## 5. Bitwise Operators

### CODE



```
main.py (Ctrl+M)
1 a = 7
2 print("For a = 7, checking given conditions are True or False:")
3 print('\na > 5 and a < 7' =>', a > 5 and a < 7)
4 print('\na > 5 or a < 7' =>', a > 5 or a < 7)
5 print('\not (a > 5 and a < 7)' =>', not(a > 5 and a < 7))
```

### OUTPUT

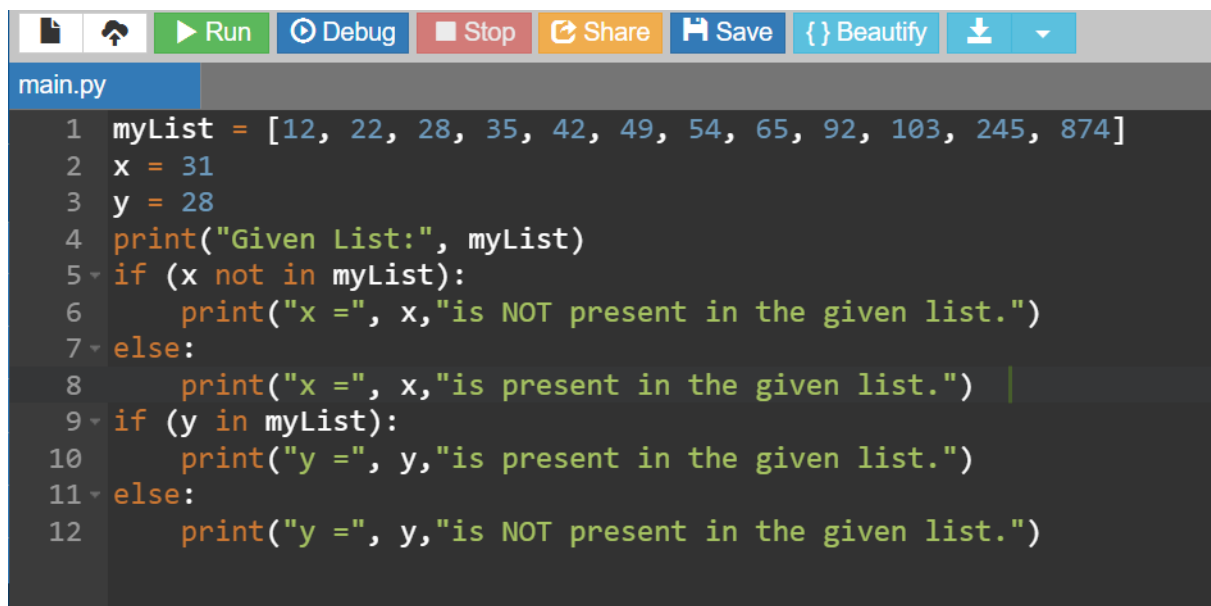


```
input
For a = 7, checking given conditions are True or False:
"a > 5 and a < 7" => False
"a > 5 or a < 7" => True
"not (a > 5 and a < 7)" => True

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

## 6. Membership Operators

### CODE



```
main.py
1 myList = [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]
2 x = 31
3 y = 28
4 print("Given List:", myList)
5 if (x not in myList):
6     print("x =", x, "is NOT present in the given list.")
7 else:
8     print("x =", x, "is present in the given list.")
9 if (y in myList):
10    print("y =", y, "is present in the given list.")
11 else:
12    print("y =", y, "is NOT present in the given list.")
```

## OUTPUT

```
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

### CODE

```
main.py
1 a = ["Rose", "Lotus"]
2 b = ["Rose", "Lotus"]
3 c = a
4 print("a is c => ", a is c)
5 print("a is not c => ", a is not c)
6 print("a is b => ", a is b)
7 print("a is not b => ", a is not b)
8 print("a == b => ", a == b)
9 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

```
import_csv_module.py > ...
1 # Importing the csv module
2 import csv
3 # Open file by passing the file path.
4 with open(r'C:\Users\Administrator\Documents\python\example.csv') as csv_file:
5     csv_read = csv.reader(csv_file, delimiter=',') # Delimiter is comma
6     count_line = 0
7     for row in csv_read:
8         if count_line == 0:
9             print(f'Column names are {", ".join(row)}')
10            count_line += 1
11        else:
12            print(f'\t{row[0]} roll number is: {row[1]} and department is: {row[2]}')
13            count_line += 1
14
15 print(f'Processed {count_line} lines.')
16
```

### OUTPUT

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Files\Python313\python.exe 'c:\Users\Administrator\.vscode\extensions\ms-python.debugpy-2024.14.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher' '59728' '--' 'c:\Users\Administrator\recipewebsite\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

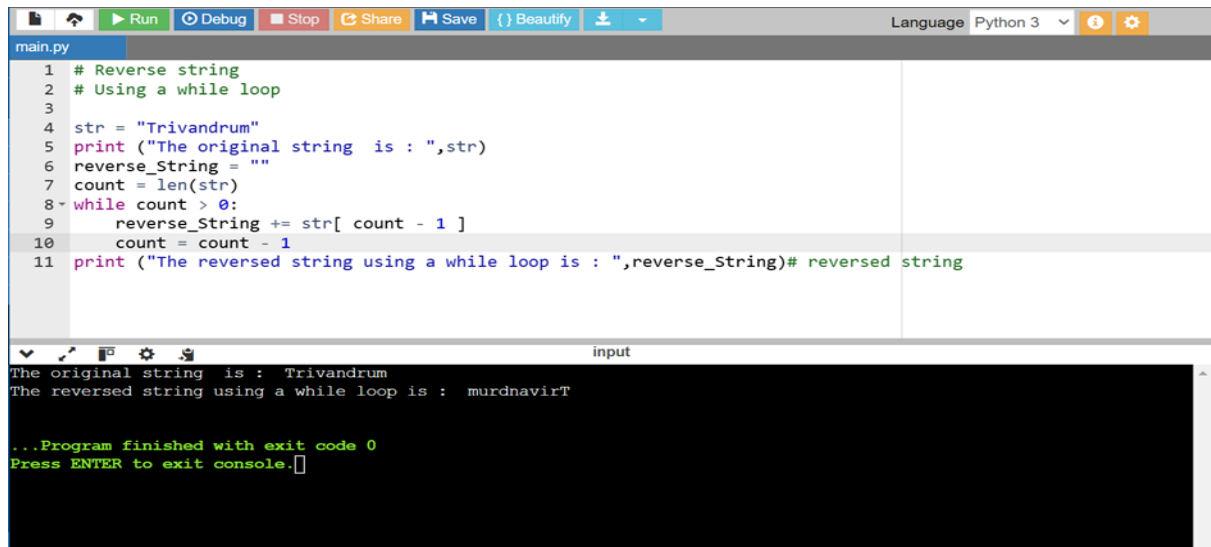
### 1. Using FOR Loop

```
main.py
1- def reverse_string(str):
2     str1 = ""
3     for i in str:
4         str1 = i + str1
5     return str1
6
7 str = "Trivandrum "
8 print("The original string is: ",str)
9 print("The reverse string is :",reverse_string(str)) # Function call

input
The original string is: Trivandrum
The reverse string is : murdnavrT

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

## 2. Using WHILE Loop



The screenshot shows a Python IDE with a file named 'main.py'. The code uses a while loop to reverse the string 'Trivandrum'. The output in the console shows the original string and the reversed string 'murdnavirT'.

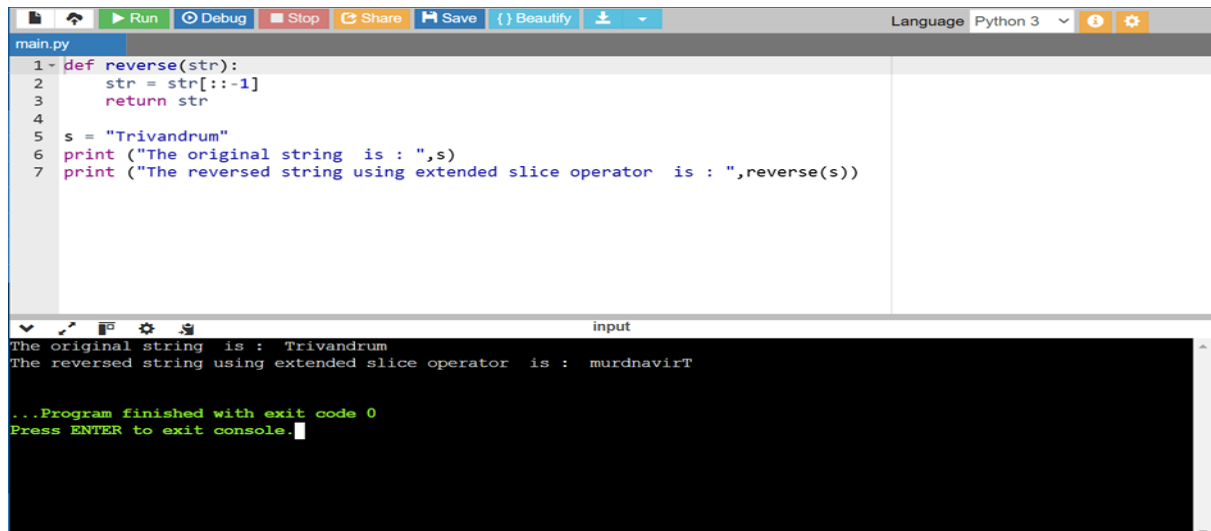
```
1 # Reverse string
2 # Using a while loop
3
4 str = "Trivandrum"
5 print ("The original string is : ",str)
6 reverse_String = ""
7 count = len(str)
8 while count > 0:
9     reverse_String += str[ count - 1 ]
10    count = count - 1
11 print ("The reversed string using a while loop is : ",reverse_String)# reversed string
```

input

```
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



The screenshot shows a Python IDE with a file named 'main.py'. A function 'reverse' is defined using the slice operator '::-1' to reverse a string. The output in the console shows the original string and the reversed string 'murdnavirT'.

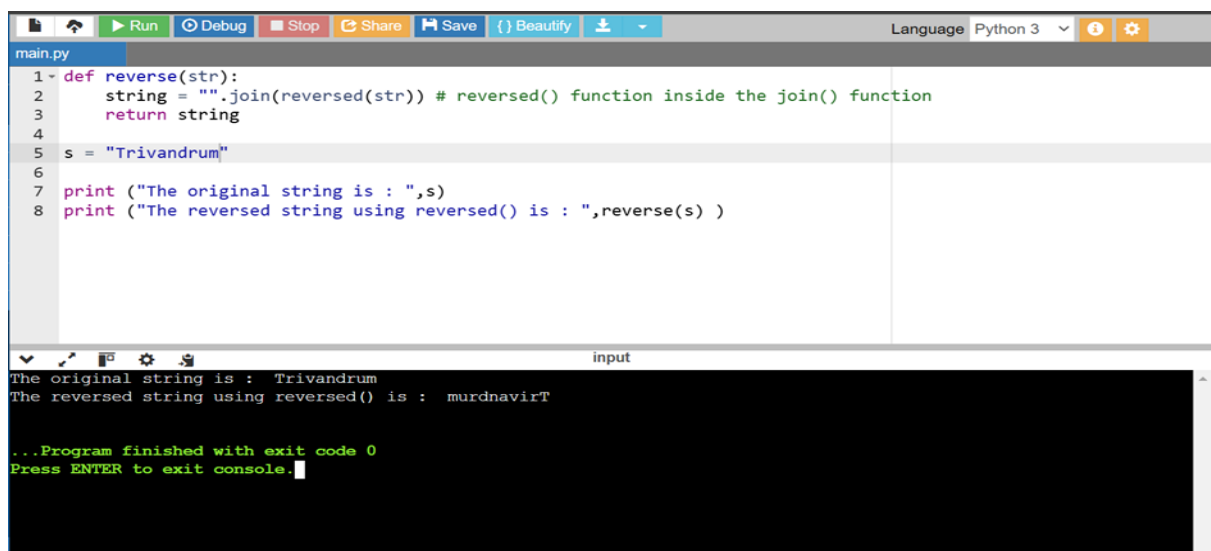
```
1 def reverse(str):
2     str = str[::-1]
3     return str
4
5 s = "Trivandrum"
6 print ("The original string is : ",s)
7 print ("The reversed string using extended slice operator is : ",reverse(s))
```

input

```
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



The screenshot shows a Python IDE with a file named 'main.py'. A function 'reverse' is defined using the 'reversed()' function and 'join()' to reverse a string. The output in the console shows the original string and the reversed string 'murdnavirT'.

```
1 def reverse(str):
2     string = "".join(reversed(str)) # reversed() function inside the join() function
3     return string
4
5 s = "Trivandrum"
6
7 print ("The original string is : ",s)
8 print ("The reversed string using reversed() is : ",reverse(s) )
```

input

```
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

```
main.py
1 def reverse(str):
2     if len(str) == 0: # Checking the length of string
3         return str
4     else:
5         return reverse(str[1:]) + str[0]
6
7 str = "Srikar"
8 print ("The original string is : ", str)
9 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:

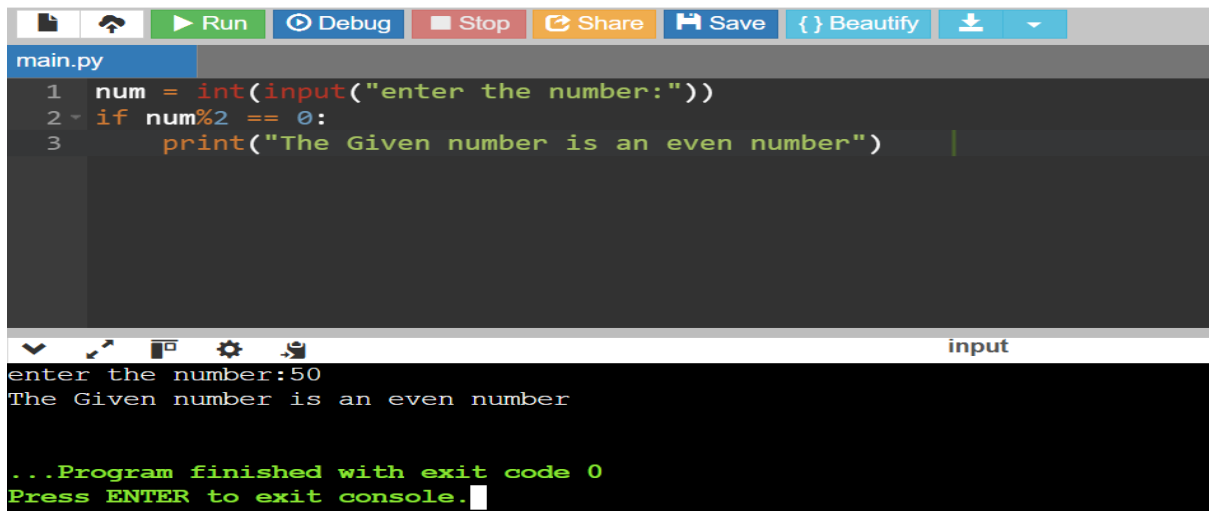
```
main.py
1 a = int (input("Enter a: "));
2 b = int (input("Enter b: "));
3 c = int (input("Enter c: "));
4 if a>b and a>c:
5     print ("From the above three numbers given a is largest");
6 if b>a and b>c:
7     print ("From the above three numbers given b is largest");
8 if c>a and c>b:
9     print ("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:

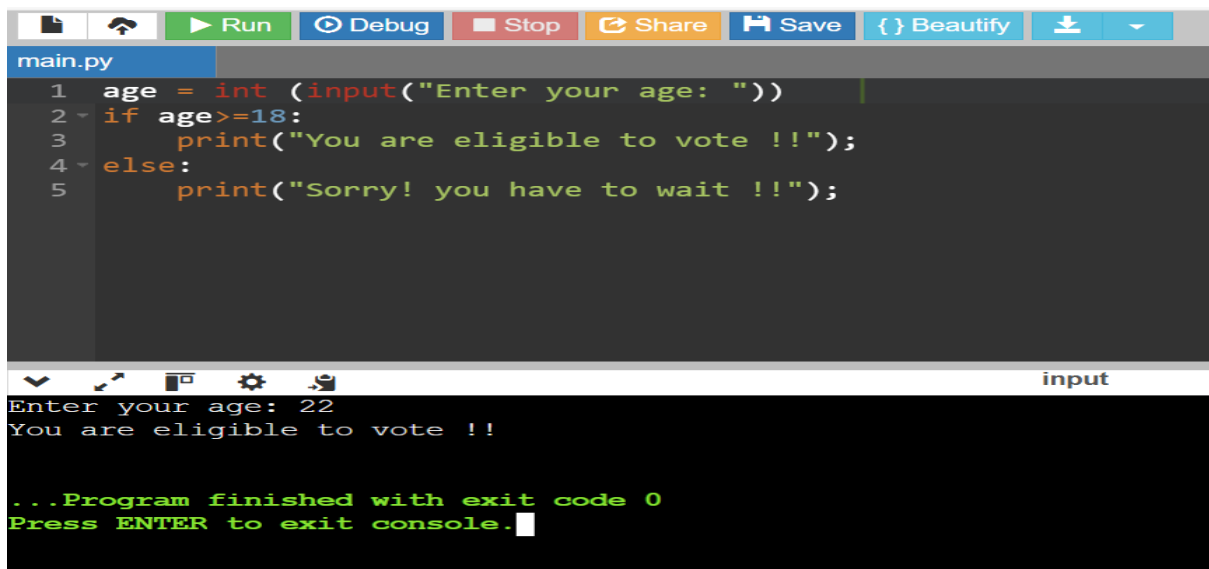


```
main.py
1 num = int(input("enter the number:"))
2 if num%2 == 0:
3     print("The Given number is an even number")

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

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

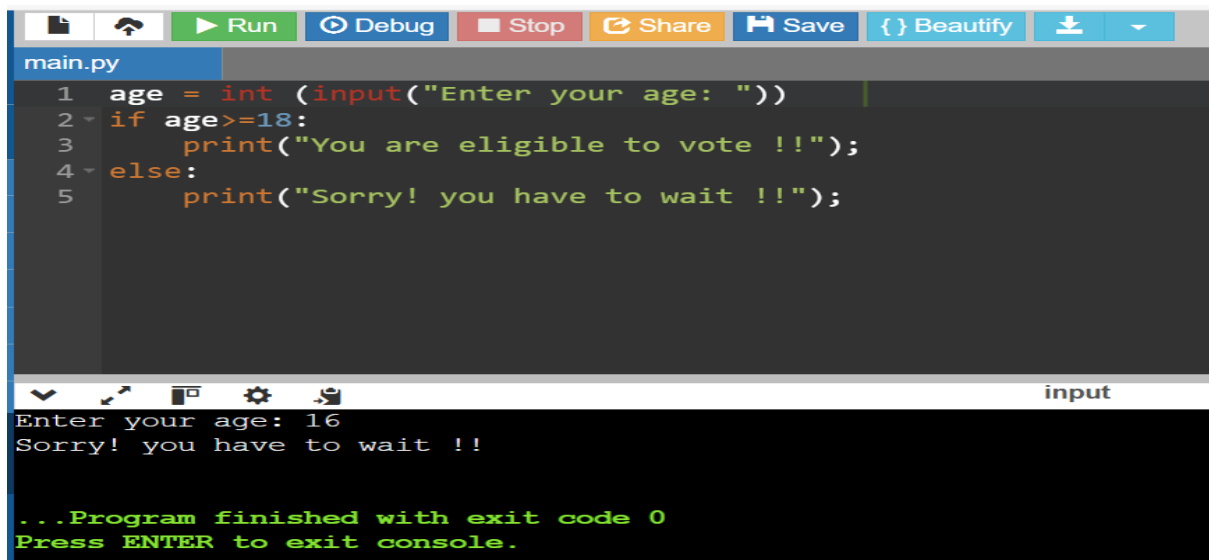
## If-Else Statement:



```
main.py
1 age = int (input("Enter your age: "))
2 if age>=18:
3     print("You are eligible to vote !!");
4 else:
5     print("Sorry! you have to wait !!");

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

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

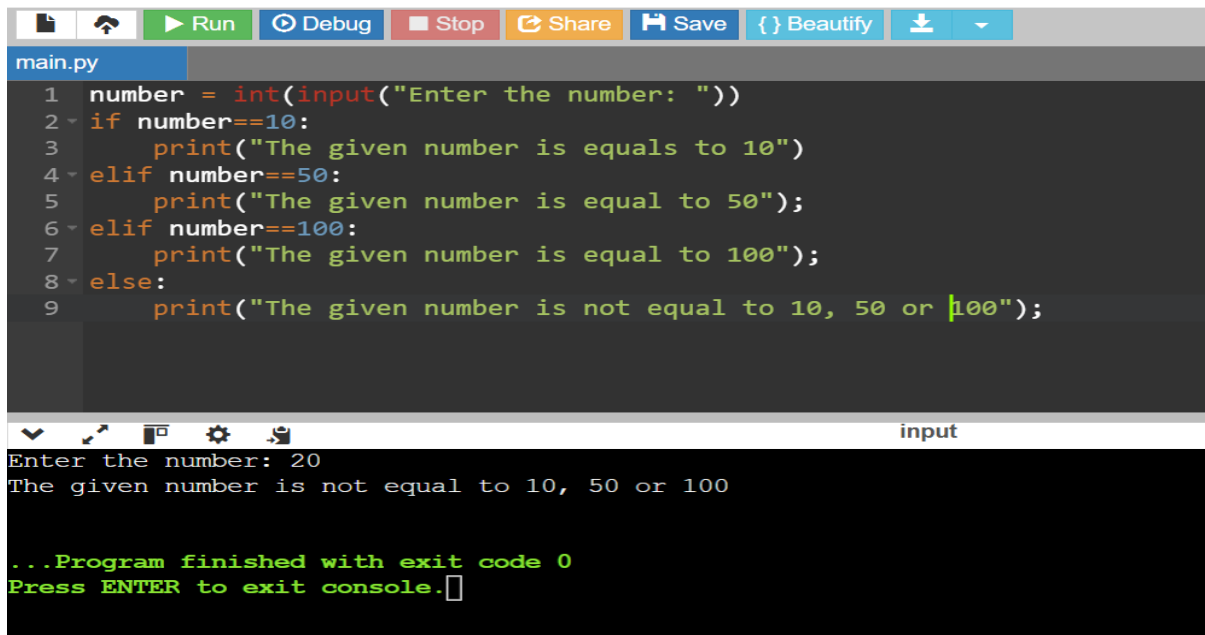


```
main.py
1 age = int (input("Enter your age: "))
2 if age>=18:
3     print("You are eligible to vote !!");
4 else:
5     print("Sorry! you have to wait !!");

input
Enter your age: 16
Sorry! you have to wait !!

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

## Elif Statement:



The screenshot shows a Python IDE with a toolbar at the top containing icons for file operations, running, debugging, stopping, sharing, saving, beautifying, and downloading. The editor window, titled 'main.py', contains the following code:

```
1 number = int(input("Enter the number: "))
2 if number==10:
3     print("The given number is equals to 10")
4 elif number==50:
5     print("The given number is equal to 50");
6 elif number==100:
7     print("The given number is equal to 100");
8 else:
9     print("The given number is not equal to 10, 50 or 100");
```

The output console at the bottom shows the program's execution. It prompts 'Enter the number: 20' and prints 'The given number is not equal to 10, 50 or 100'. The console also indicates that the program finished with exit code 0 and prompts the user to press ENTER to exit the console.

## FOR Loops:

### 1. Iterating by using index of sequence

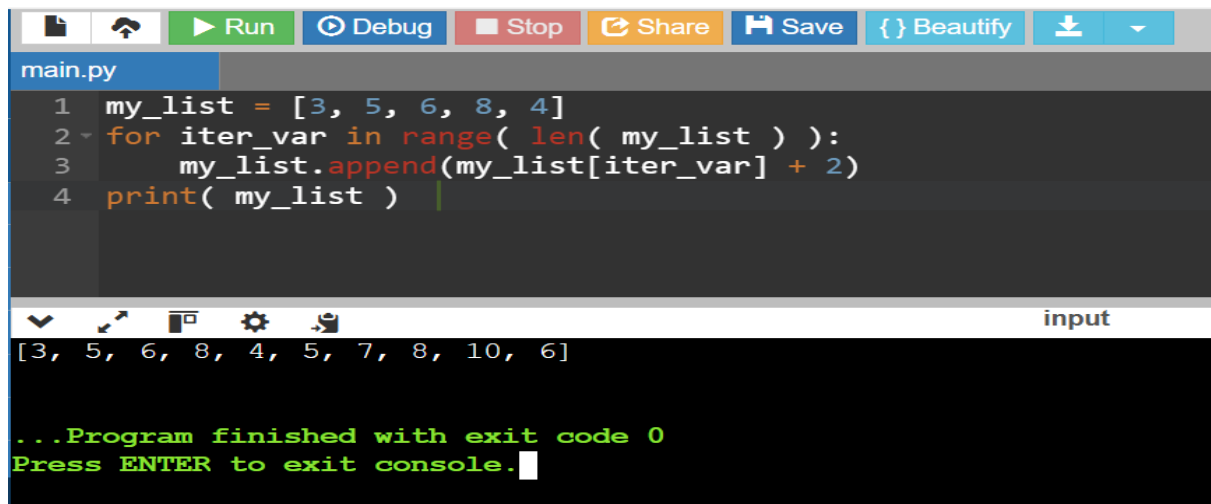


The screenshot shows a Python IDE with a toolbar at the top containing icons for file operations, running, debugging, stopping, sharing, saving, beautifying, and downloading. The editor window, titled 'main.py', contains the following code:

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

The output console at the bottom shows the program's execution. It prints 'The sum of squares is: 774'. The console also indicates that the program finished with exit code 0 and prompts the user to press ENTER to exit the console.

## 2. Using Range ()



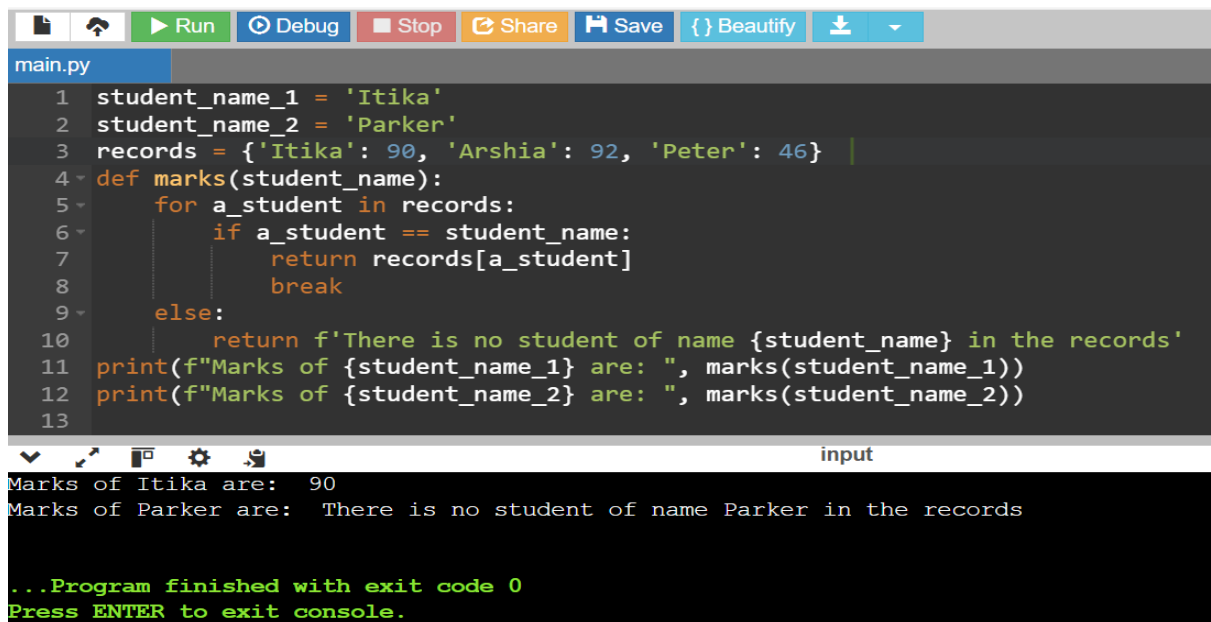
```
main.py
1 my_list = [3, 5, 6, 8, 4]
2 for iter_var in range( len( my_list ) ):
3     my_list.append(my_list[iter_var] + 2)
4 print( my_list )
```

input

[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



```
main.py
1 student_name_1 = 'Itika'
2 student_name_2 = 'Parker'
3 records = {'Itika': 90, 'Arshia': 92, 'Peter': 46}
4 def marks(student_name):
5     for a_student in records:
6         if a_student == student_name:
7             return records[a_student]
8             break
9     else:
10        return f'There is no student of name {student_name} in the records'
11 print(f'Marks of {student_name_1} are: ", marks(student_name_1))
12 print(f'Marks of {student_name_2} are: ", marks(student_name_2))
13
```

input

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



```
main.py
1 import random
2 numbers = [ ]
3 for val in range(0, 11):
4     numbers.append( random.randint( 0, 11 ) )
5 for num in range( 0, 11 ):
6     for i in numbers:
7         if num == i:
8             print( num, end = " " )
```

input

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



The screenshot shows a Python IDE with a file named 'main.py'. The code defines a variable 'num' as 21, initializes 'summation' to 0 and 'c' to 1. A while loop runs as long as 'c' is less than or equal to 'num'. Inside the loop, 'summation' is updated to 'c\*\*2 + summation', 'c' is incremented by 1, and a print statement outputs the current summation. The console shows the output 'The sum of squares is 3311' and a message indicating the program finished with exit code 0.

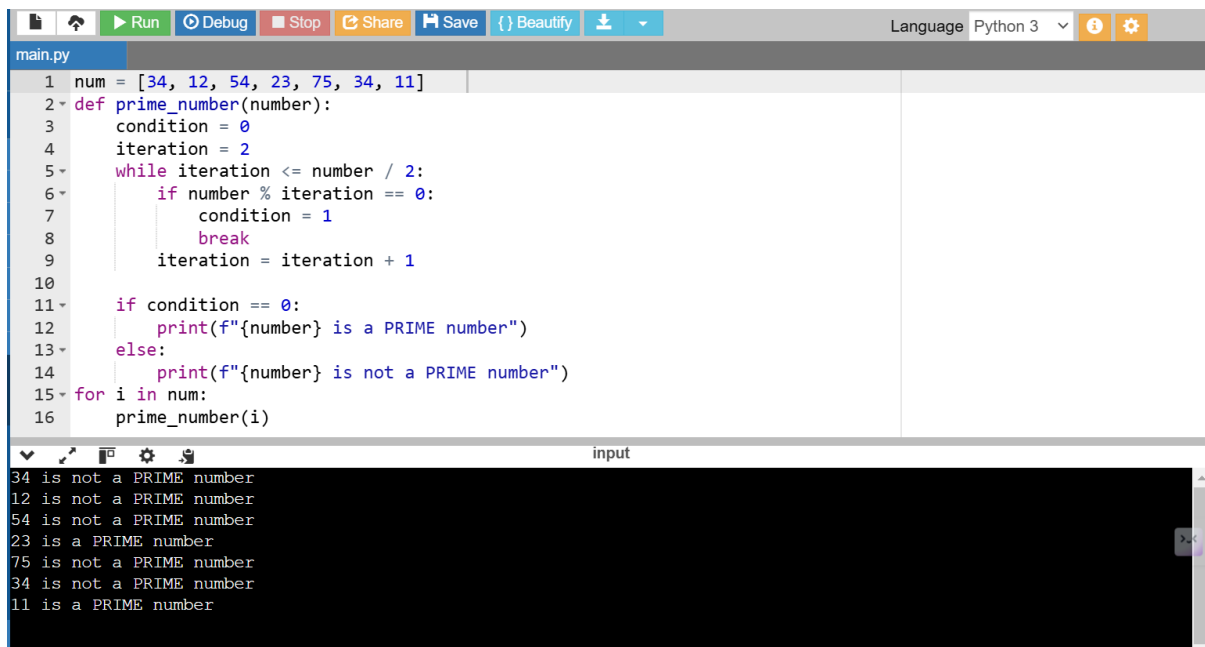
```
1 num = 21
2 summation = 0
3 c = 1
4
5 while c <= num:
6     summation = c**2 + summation
7     c = c + 1
8     print("The sum of squares is", summation)
```

input

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



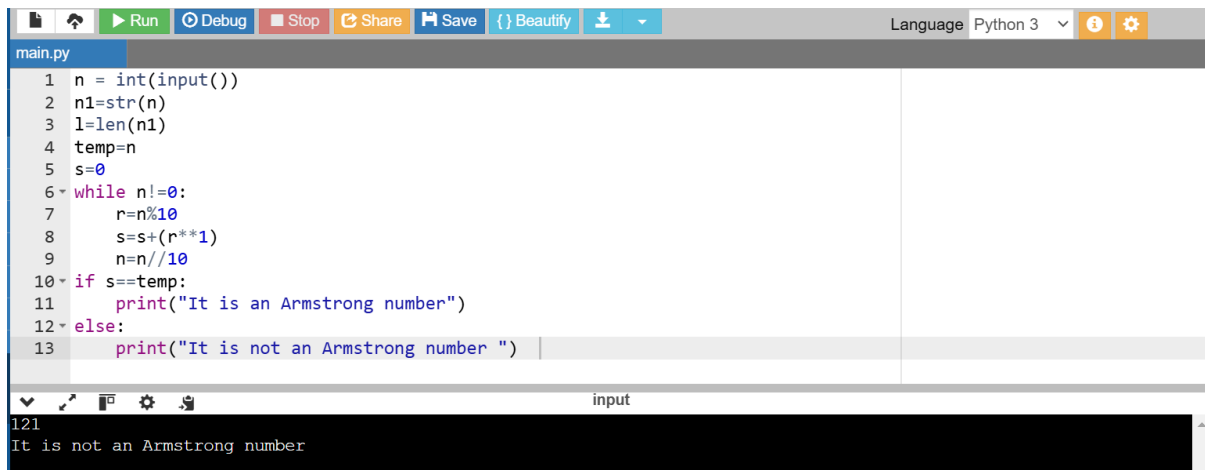
The screenshot shows a Python IDE with a file named 'main.py'. The code defines a list 'num' containing [34, 12, 54, 23, 75, 34, 11]. A function 'prime\_number' is defined, which takes a number and checks if it is prime by testing divisibility from 2 up to 'number / 2'. If a divisor is found, it sets 'condition' to 1 and breaks the loop. If 'condition' remains 0, it prints that the number is a prime. A for loop iterates over the 'num' list, calling 'prime\_number' for each element. The console shows the results for each number in the list.

```
1 num = [34, 12, 54, 23, 75, 34, 11]
2 def prime_number(number):
3     condition = 0
4     iteration = 2
5     while iteration <= number / 2:
6         if number % iteration == 0:
7             condition = 1
8             break
9         iteration = iteration + 1
10
11     if condition == 0:
12         print(f"{number} is a PRIME number")
13     else:
14         print(f"{number} is not a PRIME number")
15 for i in num:
16     prime_number(i)
```

input

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



The screenshot shows a Python IDE with a file named 'main.py'. The code takes an input 'n', converts it to a string 'n1', and gets its length 'l'. It then enters a while loop where it calculates the sum of the cubes of the digits of 'n'. If the sum 's' equals the original number 'n', it prints 'It is an Armstrong number'. Otherwise, it prints 'It is not an Armstrong number'. The console shows the input '121' and the output 'It is not an Armstrong number'.

```
1 n = int(input())
2 n1=str(n)
3 l=len(n1)
4 temp=n
5 s=0
6 while n!=0:
7     r=n%10
8     s=s+(r**1)
9     n=n//10
10 if s==temp:
11     print("It is an Armstrong number")
12 else:
13     print("It is not an Armstrong number ")
```

input

121  
It is not an Armstrong number

## 4. Multiplication Table



The screenshot shows a Python IDE with a file named `main.py`. The code defines a variable `num = 21` and a `while` loop that prints a multiplication table for `num` from `counter = 1` to `10`. The output window shows the resulting table and a message indicating the program finished with exit code 0.

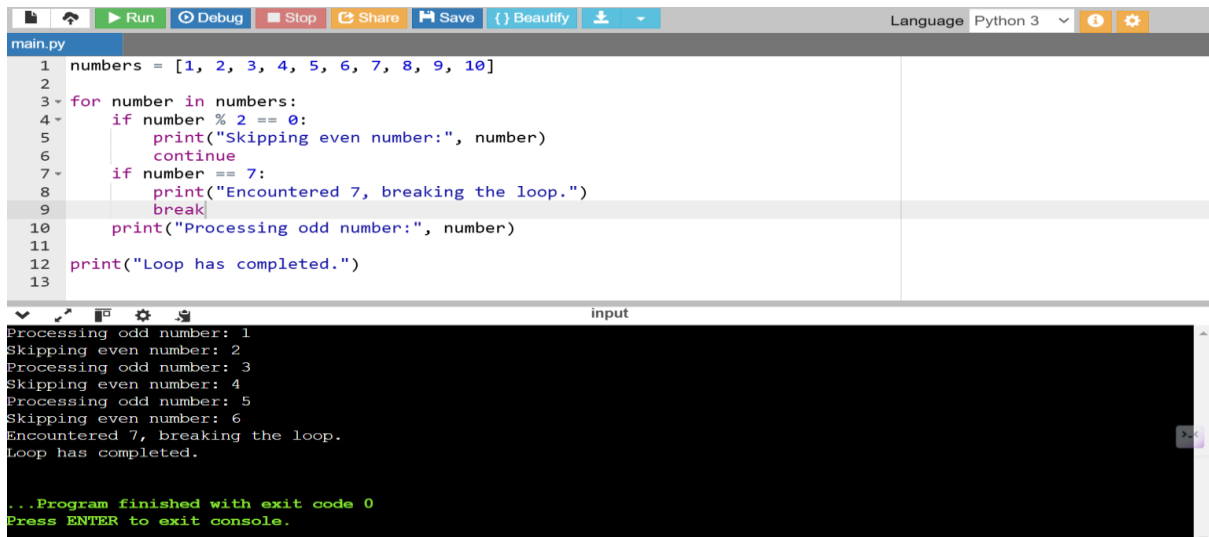
```
main.py
1 num = 21
2 counter = 1
3 print("The Multiplication Table of: ", num)
4 while counter <= 10:
5     ans = num * counter
6     print (num, 'x', counter, '=', ans)
7     counter += 1
```

input

```
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:



The screenshot shows a Python IDE with a file named `main.py`. The code uses a `for` loop to iterate over a list of numbers from 1 to 10. It skips even numbers and breaks the loop when it encounters the number 7. The output window shows the execution flow, including the break statement being triggered.

```
main.py
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
3 for number in numbers:
4     if number % 2 == 0:
5         print("Skipping even number:", number)
6         continue
7     if number == 7:
8         print("Encountered 7, breaking the loop.")
9         break
10    print("Processing odd number:", number)
11
12 print("Loop has completed.")
13
```

input

```
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:



The screenshot shows a Python IDE with a file named `main.py`. The code uses a `for` loop to iterate over the range from 10 to 21. It skips the iteration where the iterator is 15 and prints the iterator value for all other iterations. The output window shows the sequence of numbers printed, with 15 being skipped.

```
main.py
1 for iterator in range(10, 21):
2     if iterator == 15:
3         continue
4     print( iterator )
```

input

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

## STRINGS:

### 1. Creating a String in Python

```
main.py
1 str1 = 'Hello Python'
2 print(str1)
3 #Using double quotes
4 str2 = "Hello Python"
5 print(str2)
6 str3 = '''Triple quotes are generally used for
7         represent the multiline or
8         docstring'''
9 print(str3)
```

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

### 2. String Indexing

```
main.py
1 str = "JAVATPOINT"
2 print(str[0:])
3 print(str[1:5])
4 print(str[2:4])
5 print(str[:3])
6 print(str[4:7])
```

JAVATPOINT  
AVAT  
VA  
JAV  
TPO

### 3. String Splitting

```
main.py
1 str = 'JAVATPOINT'
2 print(str[-1])
3 print(str[-3])
4 print(str[-2:])
5 print(str[-4:-1])
6 print(str[-7:-2])
7 print(str[::-1])
```

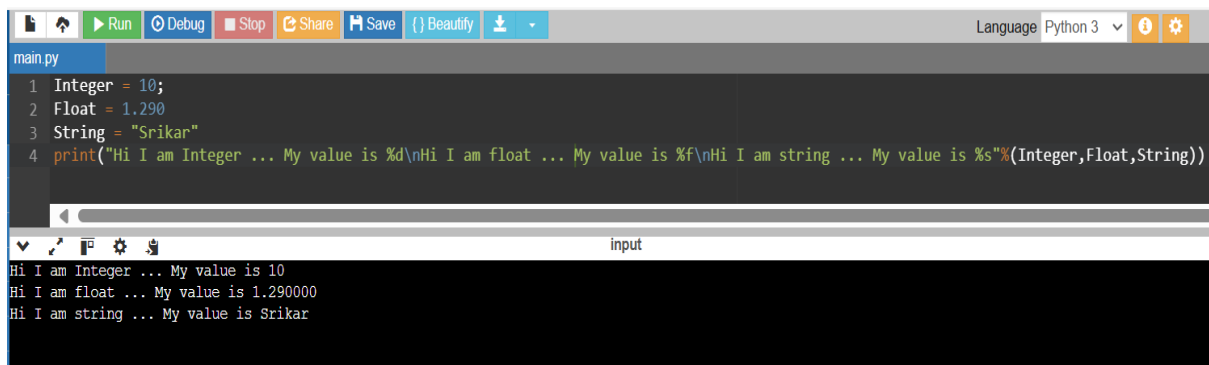
T  
I  
NT  
OIN  
ATPOI  
TNIOPTAVAJ

### 4. Python String operators

```
main.py
1 str = "Hello"
2 str1 = " world"
3 print(str*3)
4 print(str+str1)
5 print(str[4])
6 print(str[2:4]);
7 print('w' in str)
8 print('wo' not in str1)
9 print(r'C://python37')
10 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



The screenshot shows a Python IDE with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, Beautify, and a dropdown menu. The language is set to Python 3. The code in the editor is as follows:

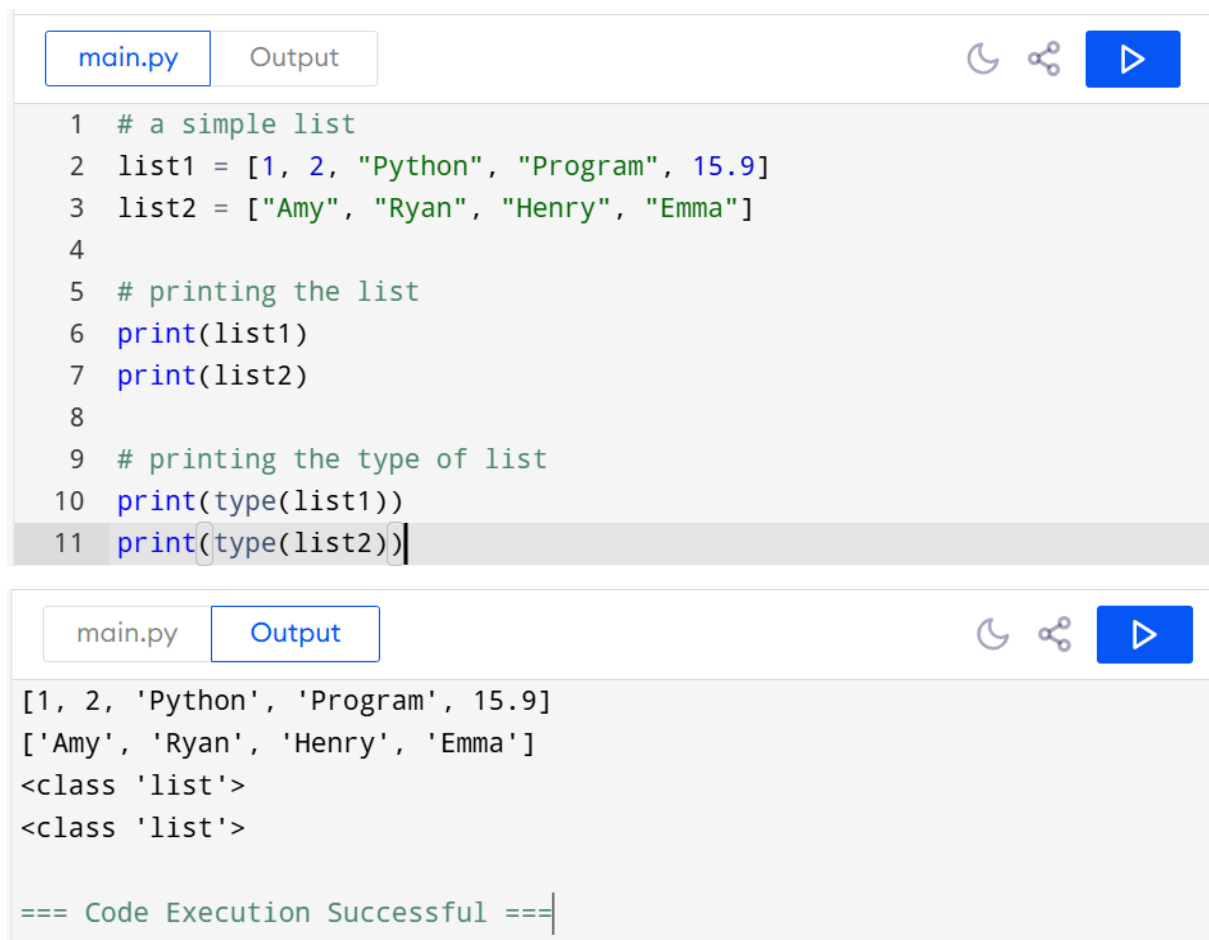
```
main.py
1 Integer = 10;
2 Float = 1.290
3 String = "Srikar"
4 print("Hi I am Integer ... My value is %d\nHi I am float ... My value is %f\nHi I am string ... My value is %s"%(Integer,Float,String))
```

Below the code editor, the output is displayed in a black console window:

```
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



The screenshot shows a Python IDE with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, Beautify, and a dropdown menu. The language is set to Python 3. The code in the editor is as follows:

```
main.py Output
1 # a simple list
2 list1 = [1, 2, "Python", "Program", 15.9]
3 list2 = ["Amy", "Ryan", "Henry", "Emma"]
4
5 # printing the list
6 print(list1)
7 print(list2)
8
9 # printing the type of list
10 print(type(list1))
11 print(type(list2))
```

Below the code editor, the output is displayed in a black console window:

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

At the bottom of the console, the message "=== Code Execution Successful ===" is displayed.






## 2. Ordered List Checking

### Example 1:

main.py




Output



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

main.py

Output






False

=== Code Execution Successful ===

### Example 2:

main.py




Output



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

main.py

Output






True

=== Code Execution Successful ===

## 3. List Indexing and Splitting

main.py




Output



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

main.py

Output






```
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

main.py



Output



```
1 list_ = [4, 5, 7, 1, 7]
2 tuple_ = (4, 1, 8, 3, 9)
3
4 print("List is: ", list_)
5 print("Tuple is: ", tuple_)
```

main.py

Output






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

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

#### 5. Mutable List vs Immutable Tuple

main.py




Output



```
1 list_ = ["Python", "Lists", "Tuples", "Differences"]
2 tuple_ = ("Python", "Lists", "Tuples", "Differences")
3
4 # modifying the last string in both data structures
5 list_[3] = "Mutable"
6 print( list_ )
7 try:
8     tuple_[3] = "Immutable"
9     print( tuple_ )
10 except TypeError:
11     print( "Tuples cannot be modified because they are immutable" )
```

main.py

Output



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

## 6. Size Difference

main.py




Output



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

main.py











Output



```
Size of tuple: 56  
Size of list: 72  
  
=== Code Execution Successful ===
```

# PYTHON FUNCTIONS


## 1. Calling a function



main.py

New File (Ctrl+M)

```
1 # Defining a function  
2 def a_function( string ):  
3     "This prints the value of length of string"  
4     return len(string)  
5 print( "Length of the string Functions is: ", a_function( "Functions" ) )  
6 print( "Length of the string Python is: ", a_function( "Python" ) )
```

input

```
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

```
1 File
2 def square( item_list ):
3     '''This function will find the square of items in the list'''
4     squares = [ ]
5     for l in item_list:
6         squares.append( l**2 )
7     return squares
8 my_list = [17, 52, 8];
9 my_result = square( my_list )
10 print( "Squares of the list are: ", my_result )
```

input

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

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

## FUNCTION ARGUMENTS

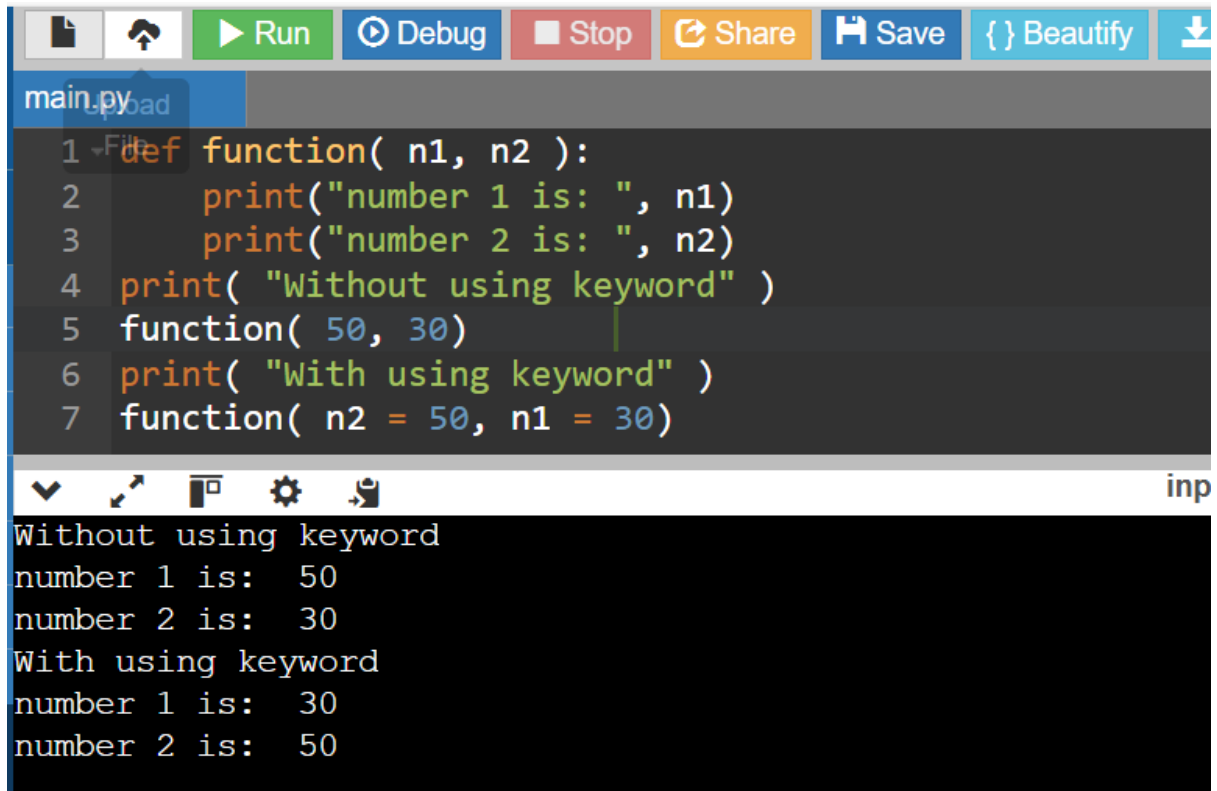
### 1. Default arguments

```
1 def function( n1, n2 = 20 ):
2     print("number 1 is: ", n1)
3     print("number 2 is: ", n2)
4 print( "Passing only one argument" )
5 function(30)
6 print( "Passing two arguments" )
7 function(50,30)
```

input

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



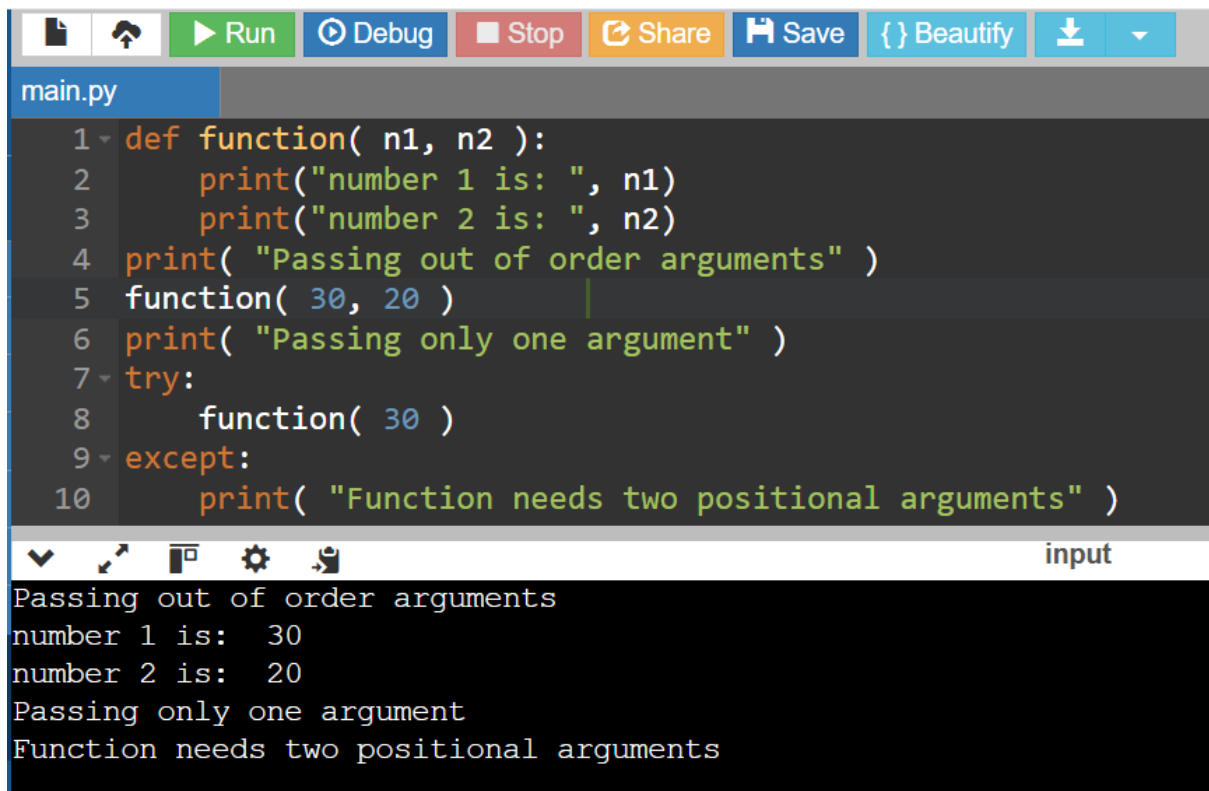
The screenshot shows a Python IDE with a toolbar at the top containing icons for file operations, running, debugging, stopping, sharing, saving, and beautifying code. The editor window, titled 'main.py', contains the following Python code:

```
1 def function( n1, n2 ):
2     print("number 1 is: ", n1)
3     print("number 2 is: ", n2)
4     print( "Without using keyword" )
5     function( 50, 30)
6     print( "With using keyword" )
7     function( n2 = 50, n1 = 30)
```

Below the editor is a console window showing the output of the code:

```
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



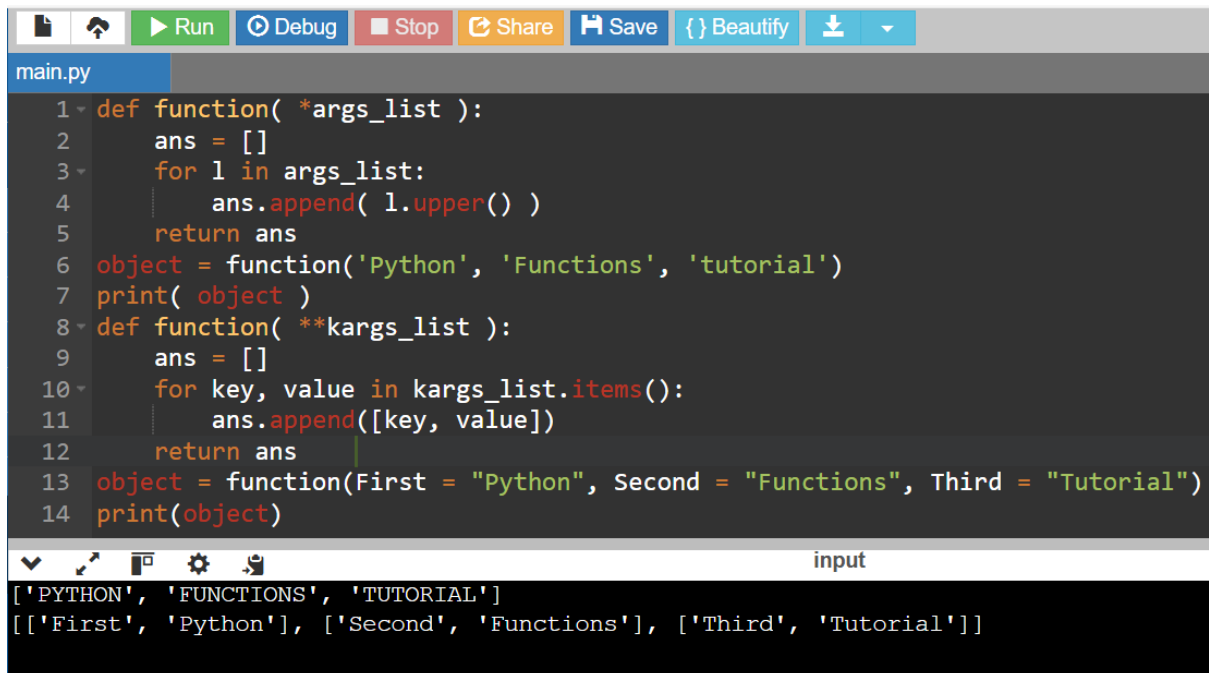
The screenshot shows a Python IDE with a toolbar at the top. The editor window, titled 'main.py', contains the following Python code:

```
1 def function( n1, n2 ):
2     print("number 1 is: ", n1)
3     print("number 2 is: ", n2)
4     print( "Passing out of order arguments" )
5     function( 30, 20 )
6     print( "Passing only one argument" )
7     try:
8         function( 30 )
9     except:
10        print( "Function needs two positional arguments" )
```

Below the editor is a console window showing the output of the code:

```
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

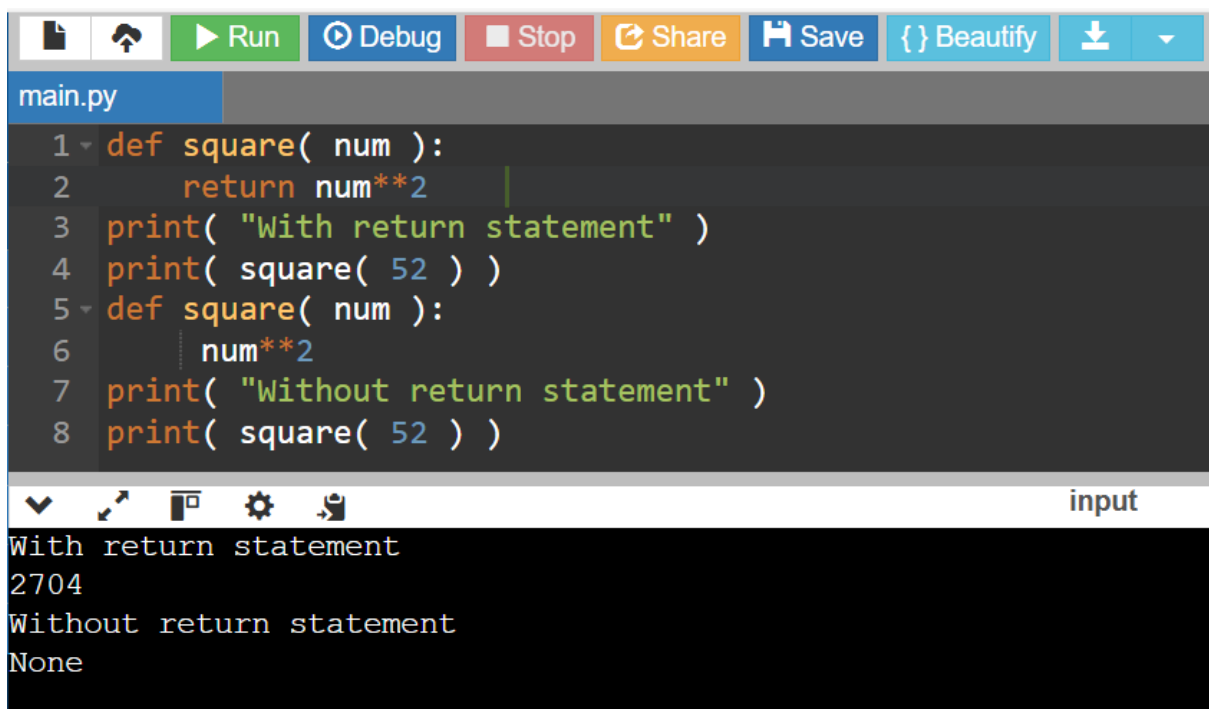


```
main.py
1 def function( *args_list ):
2     ans = []
3     for l in args_list:
4         ans.append( l.upper() )
5     return ans
6 object = function('Python', 'Functions', 'tutorial')
7 print( object )
8 def function( **kargs_list ):
9     ans = []
10    for key, value in kargs_list.items():
11        ans.append([key, value])
12    return ans
13 object = function(First = "Python", Second = "Functions", Third = "Tutorial")
14 print(object)
```

input

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

#### RETURN STATEMENT



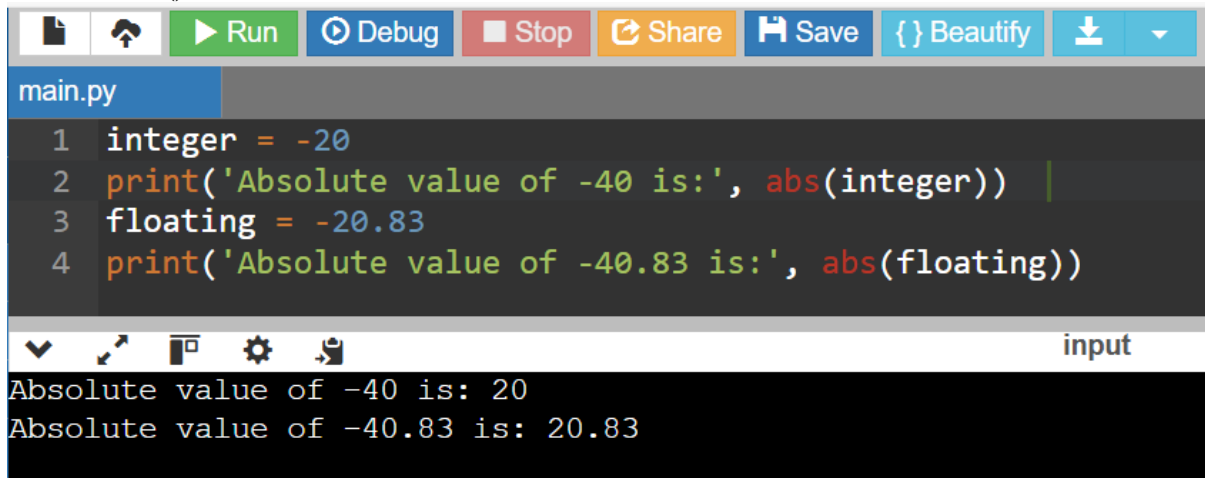
```
main.py
1 def square( num ):
2     return num**2
3 print( "With return statement" )
4 print( square( 52 ) )
5 def square( num ):
6     num**2
7 print( "Without return statement" )
8 print( square( 52 ) )
```

input

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

## PYTHON BUILT-IN FUNCTIONS

### 1. Abs () function



The screenshot shows a Python IDE interface. At the top, there is a toolbar with icons for file operations and buttons for 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. Below the toolbar, the file name 'main.py' is displayed. The code editor contains the following Python code:

```
1 integer = -20
2 print('Absolute value of -40 is:', abs(integer))
3 floating = -20.83
4 print('Absolute value of -40.83 is:', abs(floating))
```

Below the code editor, there is a toolbar with icons for view, run, settings, and a search icon. To the right of this toolbar is the label 'input'. The output area at the bottom displays the results of the code execution:

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

### 2. All () function



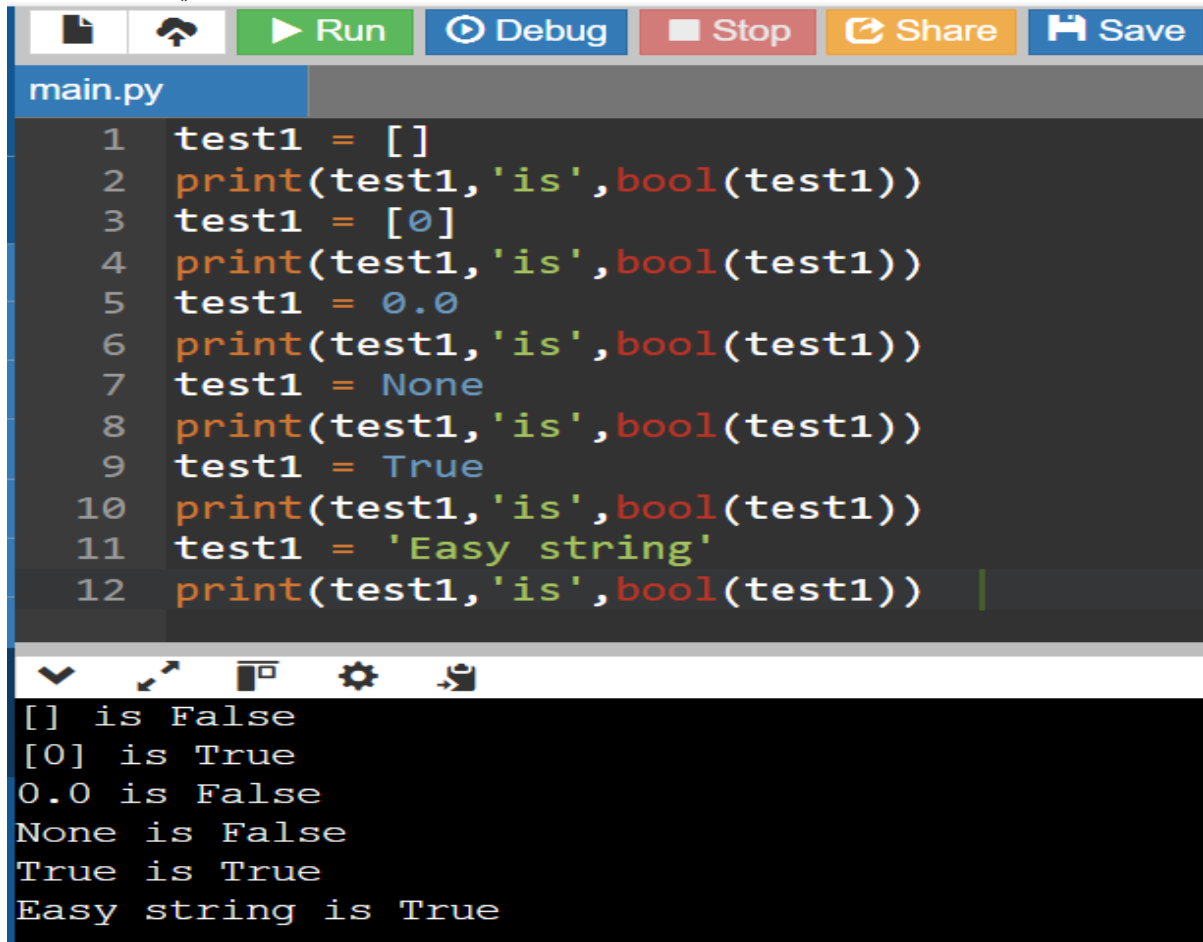
The screenshot shows a Python IDE interface. At the top, there is a toolbar with icons for file operations and buttons for 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. Below the toolbar, the file name 'main.py' is displayed. The code editor contains the following Python code:

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

Below the code editor, there is a toolbar with icons for view, run, settings, and a search icon. To the right of this toolbar is the label 'input'. The output area at the bottom displays the results of the code execution:

```
True
False
False
False
True
```

### 3. Bool () function

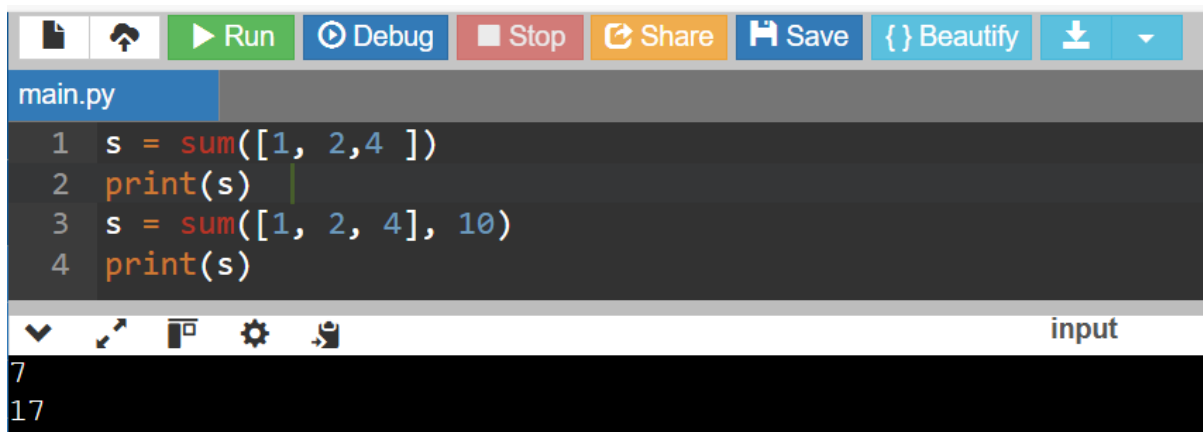


```
main.py
1 test1 = []
2 print(test1, 'is', bool(test1))
3 test1 = [0]
4 print(test1, 'is', bool(test1))
5 test1 = 0.0
6 print(test1, 'is', bool(test1))
7 test1 = None
8 print(test1, 'is', bool(test1))
9 test1 = True
10 print(test1, 'is', bool(test1))
11 test1 = 'Easy string'
12 print(test1, 'is',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

Output:

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

### 4. Sum () Function



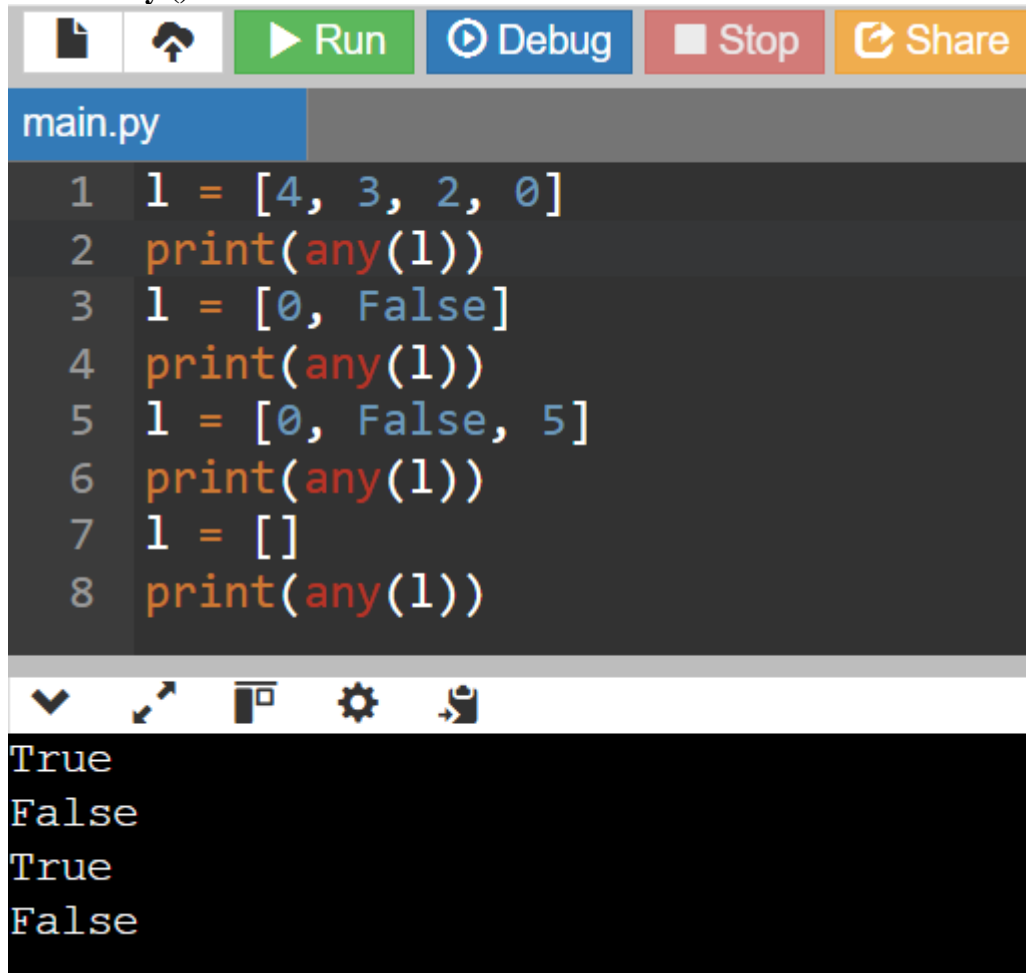
```
main.py
1 s = sum([1, 2, 4 ])
2 print(s)
3 s = sum([1, 2, 4], 10)
4 print(s)
```

Output:

```
7
17
```



## 5. Any () function



The screenshot shows a Python IDE with a toolbar at the top containing icons for file operations, a 'Run' button (green), a 'Debug' button (blue), a 'Stop' button (red), and a 'Share' button (orange). Below the toolbar, a tab labeled 'main.py' is active. The code editor contains the following Python code:

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

Below the code editor, a toolbar with icons for view, zoom, and settings is visible. The output console at the bottom displays the results of the code execution:

```
True
False
True
False
```

## PYTHON LAMBDA FUNCTION

### 1. Lambda function example



The screenshot shows a Python IDE with a toolbar at the top containing icons for file operations, a 'Run' button (green), a 'Debug' button (blue), a 'Stop' button (red), and a 'Share' button (orange). Below the toolbar, a tab labeled 'main.py' is active. The code editor contains the following Python code:

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

Below the code editor, a toolbar with icons for view, zoom, and settings is visible. The output console at the bottom displays the result of the code execution:

```
10
```

## 2. Distinction between Lambda and Def Function

```
main.py
1 def reciprocal( num ):
2     return 1 / num
3 lambda_reciprocal = lambda num: 1 / num
4 print( "Def keyword: ", reciprocal(6) )
5 print( "Lambda keyword: ", lambda_reciprocal(6) )
```

input

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

## 3. Using Lambda Function with map ()

```
main.py
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 )
```

input

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

## 4. Using Lambda Function with List

```
main.py
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 value of all numbers from 0 to 10: 100
```

## 5. Using Lambda Function with Multiple Statements

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

input

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

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