

PYTHON-BASICS-ASSIGNMENTS-1

1.Operators in python.

1. Arithmetic Operators.

main.py	Output
<pre>1 a = int(input("Enter any number : ")) 2 b = int(input("Enter any number : ")) 3 4 print("For a =", a, "and b =", b, "\nCalculate the following:") 5 6 # printing different results 7 print('Addition of two numbers: a + b =', a + b) 8 print('Subtraction of two numbers: a - b =', a - b) 9 print('Multiplication of two numbers: a * b =', a * b) 10 print('Division of two numbers: a / b =', a / b) 11 print('Floor division of two numbers: a // b =', a // b) 12 print('Reminder of two numbers: a mod b =', a % b) 13 print('Exponent of two numbers: a ^ b =', a ** b)</pre>	<pre>Enter any number : 15 Enter any number : 3 For a = 15 and b = 3 Calculate the following: Addition of two numbers: a + b = 18 Subtraction of two numbers: a - b = 12 Multiplication of two numbers: a * b = 45 Division of two numbers: a / b = 5.0 Floor division of two numbers: a // b = 5 Reminder of two numbers: a mod b = 0 Exponent of two numbers: a ^ b = 3375 === Code Execution Successful ===</pre>

Output
<pre>Enter any number : 25 Enter any number : 0 For a = 25 and b = 0 Calculate the following: Addition of two numbers: a + b = 25 Subtraction of two numbers: a - b = 25 Multiplication of two numbers: a * b = 0 ERROR! Traceback (most recent call last): File "<main.py>", line 10, in <module> ZeroDivisionError: division by zero === Code Exited With Errors ===</pre>

Output
<pre>Enter any number : -6 Enter any number : 4 For a = -6 and b = 4 Calculate the following: Addition of two numbers: a + b = -2 Subtraction of two numbers: a - b = -10 Multiplication of two numbers: a * b = -24 Division of two numbers: a / b = -1.5 Floor division of two numbers: a // b = -2 Reminder of two numbers: a mod b = 2 Exponent of two numbers: a ^ b = 1296 === Code Execution Successful ===</pre>

2. Comparison Operators.

main.py	Output
<pre>1 x = int(input("Enter any number for x: ")) 2 y = int(input("Enter any number for y: ")) 3 4 print("For x =", x, "and y =", y, "\nCheck the following comparisons:") 5 6 # printing different comparison results 7 print(f'Is {x} equal to {y}?', x == y) 8 print(f'Is {x} not equal to {y}?', x != y) 9 print(f'Is {x} less than {y}?', x < y) 10 print(f'Is {x} greater than {y}?', x > y) 11 print(f'Is {x} less than or equal to {y}?', x <= y) 12 print(f'Is {x} greater than or equal to {y}?', x >= y) 13</pre>	<pre>Enter any number for x: 45 Enter any number for y: 7 For x = 45 and y = 7 Check the following comparisons: Is 45 equal to 7? False Is 45 not equal to 7? True Is 45 less than 7? False Is 45 greater than 7? True Is 45 less than or equal to 7? False Is 45 greater than or equal to 7? True === Code Execution Successful ===</pre>

```
Output

Enter any number for x: -10
Enter any number for y: 5
For x = -10 and y = 5
Check the following comparisons:
Is -10 equal to 5? False
Is -10 not equal to 5? True
Is -10 less than 5? True
Is -10 greater than 5? False
Is -10 less than or equal to 5? True
Is -10 greater than or equal to 5? False

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

```
Output

Enter any number for x: 100
Enter any number for y: 100
For x = 100 and y = 100
Check the following comparisons:
Is 100 equal to 100? True
Is 100 not equal to 100? False
Is 100 less than 100? False
Is 100 greater than 100? False
Is 100 less than or equal to 100? True
Is 100 greater than or equal to 100? True

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

3. Assignment Operators.

main.py	Output
<pre>1 a = int(input("Enter any number for a: ")) 2 b = int(input("Enter any number for b: ")) 3 c=a 4 print("For a =", a, "and b =", b, "\nCheck the following comparisons:") 5 # printing the different results 6 a += b 7 print('a += b:', a) 8 a = c # Resetting a to its initial value 9 a -= b 10 print('a -= b:', a) 11 a = c # Resetting a to its initial value 12 a *= b 13 print('a *= b:', a) 14 a = c # Resetting a to its initial value 15 a /= b 16 print('a /= b:', a) 17 a = c # Resetting a to its initial value 18 a %= b 19 print('a %= b:', a) 20 a = c # Resetting a to its initial value 21 a **= b 22 print('a **= b:', a) 23 a = c # Resetting a to its initial value 24 a //= b 25 print('a //= b:', a) 26</pre>	<pre>Enter any number for a: 10 Enter any number for b: 5 For a = 10 and b = 5 Check the following comparisons: a += b: 15 a -= b: 5 a *= b: 50 a /= b: 2.0 a %= b: 0 a **= b: 100000 a //= b: 2 === Code Execution Successful ===</pre>

```
Output

Enter any number for a: -2
Enter any number for b: 4
For a = -2 and b = 4
Check the following comparisons:
a += b: 2
a -= b: -6
a *= b: -8
a /= b: -0.5
a %= b: 2
a **= b: 16
a //= b: -1

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

```
Output

Enter any number for a: 2
Enter any number for b: 2
For a = 2 and b = 2
Check the following comparisons:
a += b: 4
a -= b: 0
a *= b: 4
a /= b: 1.0
a %= b: 0
a **= b: 4
a //= b: 1

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

4. Logical Operators.

main.py	Output
<pre>1 a = int(input("Enter a value for a: ")) # Taking input for a 2 3 # printing different results 4 print(f"For a = {a}, checking whether the following conditions are True or False:") 5 6 print(f'a > 5 and a < 7' =>, a > 5 and a < 7) 7 print(f'a > 5 or a < 7' =>, a > 5 or a < 7) 8 print(f'not (a > 5 and a < 7)' =>, not(a > 5 and a < 7)) 9</pre>	<pre>Enter a value for a: 6 For a = 6, checking whether the following conditions are True or False: "a > 5 and a < 7" => True "a > 5 or a < 7" => True "not (a > 5 and a < 7)" => False === Code Execution Successful ===</pre>

Output	Output
<pre>Enter a value for a: 8 For a = 8, 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 === Code Execution Successful ===</pre>	<pre>Enter a value for a: 4 For a = 4, 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 === Code Execution Successful ===</pre>

5. Bitwise Operators.

main.py	Output
<pre>1 a = int(input("Enter a value for a: ")) # Taking input for a 2 b = int(input("Enter a value for b: ")) # Taking input for b 3 4 # printing different results 5 print(f'a & b : ', a & b) 6 print(f'a b : ', a b) 7 print(f'a ^ b : ', a ^ b) 8 print(f'~a : ', ~a) 9 print(f'a << b : ', a << b) 10 print(f'a >> b : ', a >> b) 11</pre>	<pre>Enter a value for a: 5 Enter a value for b: 7 a & b : 5 a b : 7 a ^ b : 2 ~a : -6 a << b : 640 a >> b : 0 === Code Execution Successful ===</pre>

Output	Output
<pre>Enter a value for a: 8 Enter a value for b: 7 a & b : 0 a b : 15 a ^ b : 15 ~a : -9 a << b : 1024 a >> b : 0 === Code Execution Successful ===</pre>	<pre>Enter a value for a: 2 Enter a value for b: 3 a & b : 2 a b : 3 a ^ b : 1 ~a : -3 a << b : 16 a >> b : 0 === Code Execution Successful ===</pre>

6. Membership Operators.

main.py	Output
<pre>1 # initializing a list 2 myList = [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874] 3 # taking input from the user 4 x = int(input("Enter a value for x: ")) 5 y = int(input("Enter a value for y: ")) 6 # printing the given list 7 print("Given List:", myList) 8 # checking if x is present in the list or not 9 if (x not in myList): 10 print(f"x = {x} is NOT present in the given list.") 11 else: 12 print(f"x = {x} is present in the given list.") 13 14 # checking if y is present in the list or not 15 if (y in myList): 16 print(f"y = {y} is present in the given list.") 17 else: 18 print(f"y = {y} is NOT present in the given list.") 19</pre>	<pre>Enter a value for x: 84 Enter a value for y: 42 Given List: [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874] x = 84 is NOT present in the given list. y = 42 is present in the given list. === Code Execution Successful ===</pre>

Output
<pre>Enter a value for x: 245 Enter a value for y: 12 Given List: [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874] x = 245 is present in the given list. y = 12 is present in the given list. === Code Execution Successful ===</pre>

Output
<pre>Enter a value for x: 10 Enter a value for y: 654 Given List: [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874] x = 10 is NOT present in the given list. y = 654 is NOT present in the given list. === Code Execution Successful ===</pre>

7. Identity Operators.

main.py	Output
<pre>1 a = ["Rose", "Lotus"] 2 b = ["Rose", "Lotus"] 3 # initializing variable c and storing the value of a in c 4 c = a 5 # printing the different results 6 print("a is c => ", a is c) 7 print("a is not c => ", a is not c) 8 print("a is b => ", a is b) 9 print("a is not b => ", a is not b) 10 print("a == b => ", a == b) 11 print("a != b => ", a != b)</pre>	<pre>a is c => True a is not c => False a is b => False a is not b => True a == b => True a != b => False === Code Execution Successful ===</pre>

2. Reversing a string in Python

1. Using for loop.

main.py	Output
<pre>1 def reverse_string(str): 2 str1 = "" # Declaring empty string to store the reversed string 3 for i in str: 4 str1 = i + str1 # Adding each character to the front of str1 5 return str1 # It will return the reversed string to the caller 6 7 str = "JavaTpoint" # Given String 8 print("The original string is: ", str) 9 print("The reverse string is: ", reverse_string(str)) # Function call 10</pre>	<pre>The original string is: JavaTpoint The reverse string is: tniopTavaJ === Code Execution Successful ===</pre>

2. Using while loop.

main.py	Output
<pre>1 # Reverse string 2 # Using a while loop 3 str = "JavaTpoint" # string variable 4 print("The original string is : ", str) 5 reverse_String = "" # Empty String 6 count = len(str) # Find length of a string and save in count variable 7 while count > 0: 8 reverse_String += str[count - 1] # Save the value of str[count-1] 9 count = count - 1 # Decrement index 10 print("The reversed string using a while loop is : ", reverse_String) 11</pre>	<pre>The original string is : JavaTpoint The reversed string using a while loop is : tniopTavaJ === Code Execution Successful ===</pre>

3. Using the slice operator.

main.py	Output
<pre>1 # Reverse a string 2 # Using slice syntax 3 # reverse(str) Function to reverse a string 4 def reverse(str): 5 str = str[::-1] # Slice operator to reverse the string 6 return str 7 8 s = "JavaTpoint" # Given string 9 print("The original string is : ", s) 10 print("The reversed string using extended slice operator is : ", 11 reverse(s))</pre>	<pre>The original string is : JavaTpoint The reversed string using extended slice operator is : tniopTavaJ === Code Execution Successful ===</pre>





4. Using the reversed() function.

main.py	Output
<pre>1 # Reverse a string using reversed() 2 # Function to reverse a string 3 def reverse(str): 4 string = "".join(reversed(str)) # reversed() function inside the join() function 5 return string 6 7 s = "JavaTpoint" # Given string 8 9 print("The original string is : ", s) 10 print("The reversed string using reversed() is : ", reverse(s))</pre>	<pre>The original string is : JavaTpoint The reversed string using reversed() is : tniopTavaJ === Code Execution Successful ===</pre>

5. Using the recursion.

main.py	Output
<pre>1 # Reverse a string 2 # Using recursion 3 def reverse(str): 4 if len(str) == 0: # Checking the length of the string 5 return str 6 else: 7 return reverse(str[1:]) + str[0] # Recursion step 8 9 str = "Devansh Sharma" # Given string 10 print("The original string is : ", str) 11 print("The reversed string (using recursion) is : ", reverse(str))</pre>	<pre>The original string is : Devansh Sharma The reversed string (using recursion) is : amrahS hsnaveD === Code Execution Successful ===</pre>

3. Reading CSV file in Python:



Run >

Result Size: 601 x 407

Get your own Python server

Python code





data.csv

```
import csv
with open(r'data.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'The CSV file contains \n \n{',
".join(row)}')
        else:
            print(f'{row[0]} \t{row[1]} \t{row[2]}')
            count_line+=1
    print(f'\nProcessed {count_line} lines.')
```

The CSV file contains

Name	Roll no	Department
Yonith	01	IT
Ram	43	CSE
Satish	3	Civil
Sai	7	EEE

Processed 5 lines.



Run >

Result Size: 601 x 407

Get your own Python server

Python code

data.csv

```
Name,Roll no,Department
Yonith,01,IT
Ram,43,CSE
Satish,3,Civil
Sai,7,EEE
```

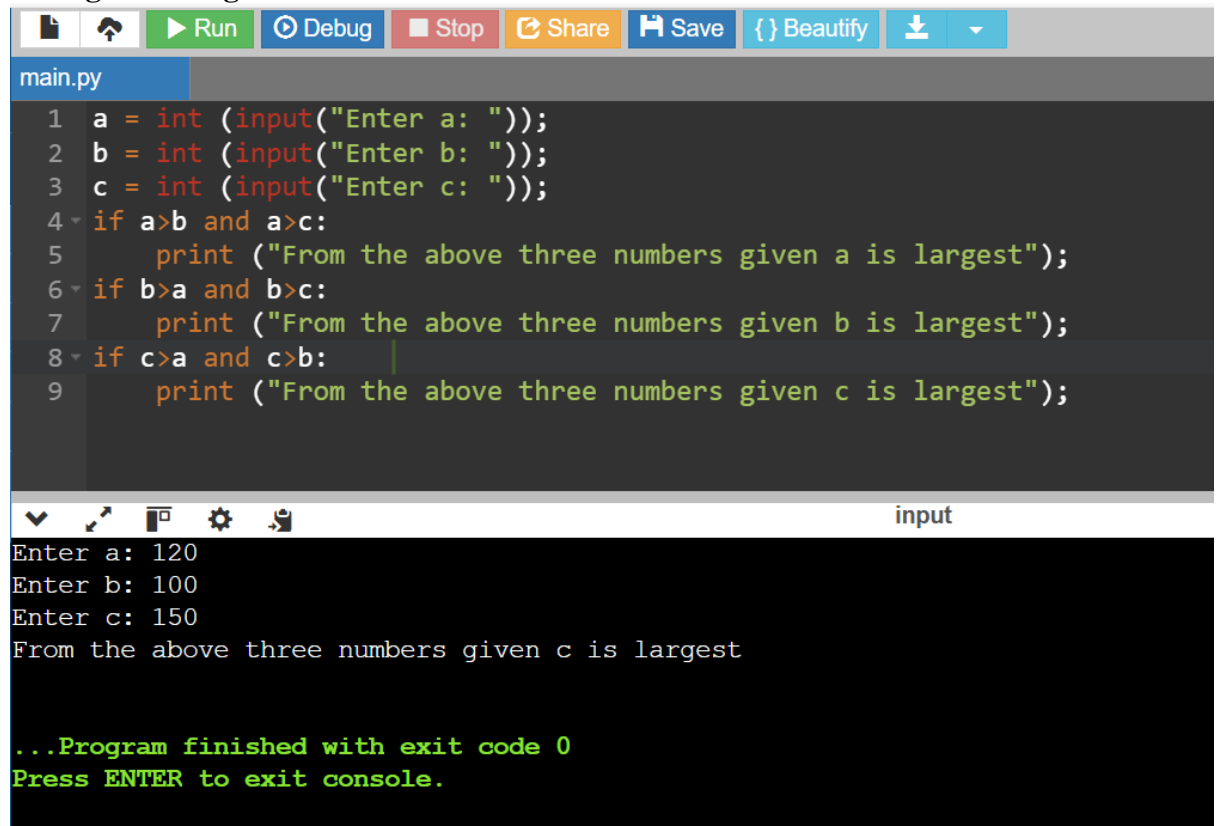
The CSV file contains

Name	Roll no	Department
Yonith	01	IT
Ram	43	CSE
Satish	3	Civil
Sai	7	EEE

Processed 5 lines.

4. Python If Statement:

1. Largest among Three numbers`



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 file name is 'main.py'. The code is as follows:

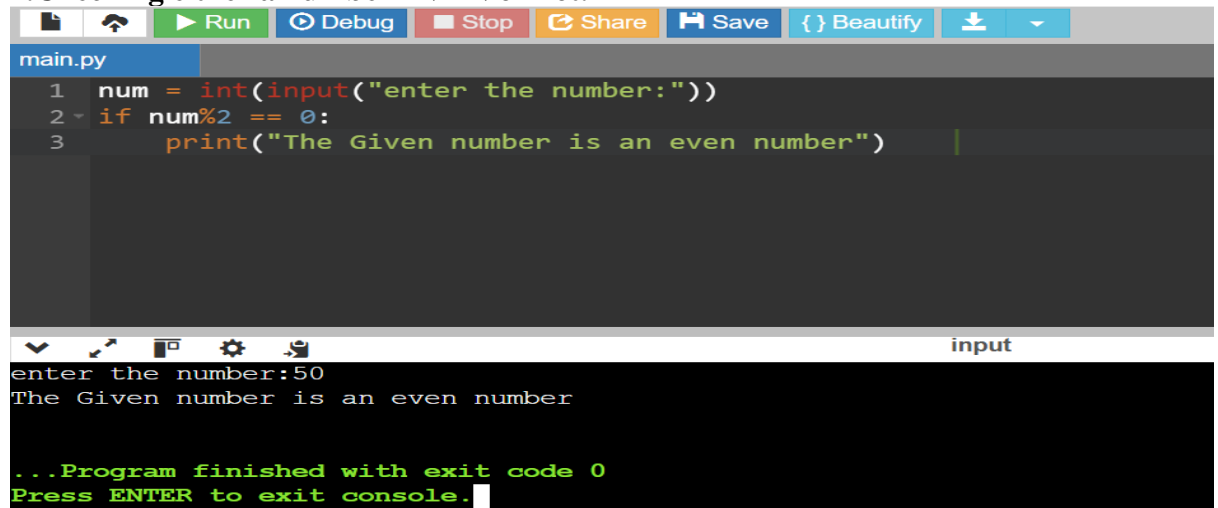
```
1 a = int (input("Enter a: "));
2 b = int (input("Enter b: "));
3 c = int (input("Enter c: "));
4 if a>b and a>c:
5     print ("From the above three numbers given a is largest");
6 if b>a and b>c:
7     print ("From the above three numbers given b is largest");
8 if c>a and c>b:
9     print ("From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

The console output shows the program execution with user input:

```
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.
```

2. Checking either a number EVEN or not.



The screenshot shows a Python IDE with a toolbar at the top. The file name is 'main.py'. The code is as follows:

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

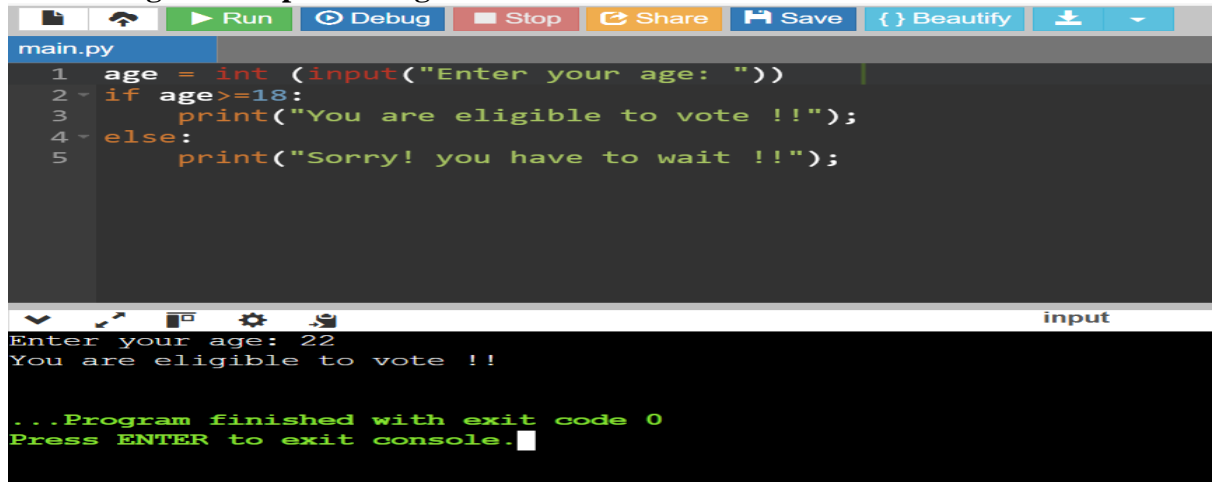
The console output shows the program execution with user input:

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

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


Python If-Else Statement:

1. Checking either a person eligible to vote or not.

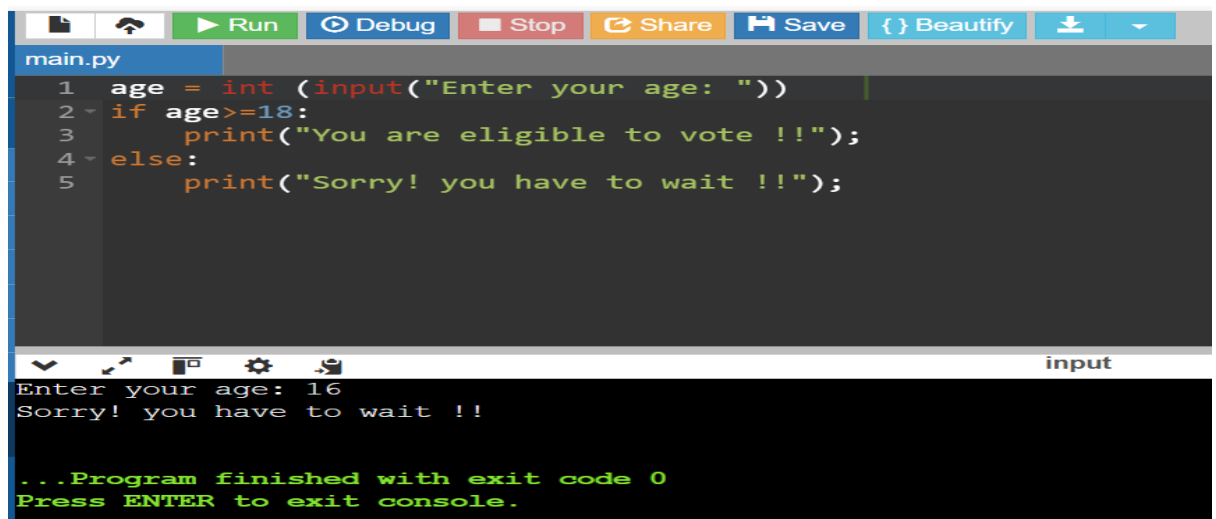


```
main.py
1 age = int (input("Enter your age: "))
2 if age>=18:
3     print("You are eligible to vote !!");
4 else:
5     print("Sorry! you have to wait !!");
```

input

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

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



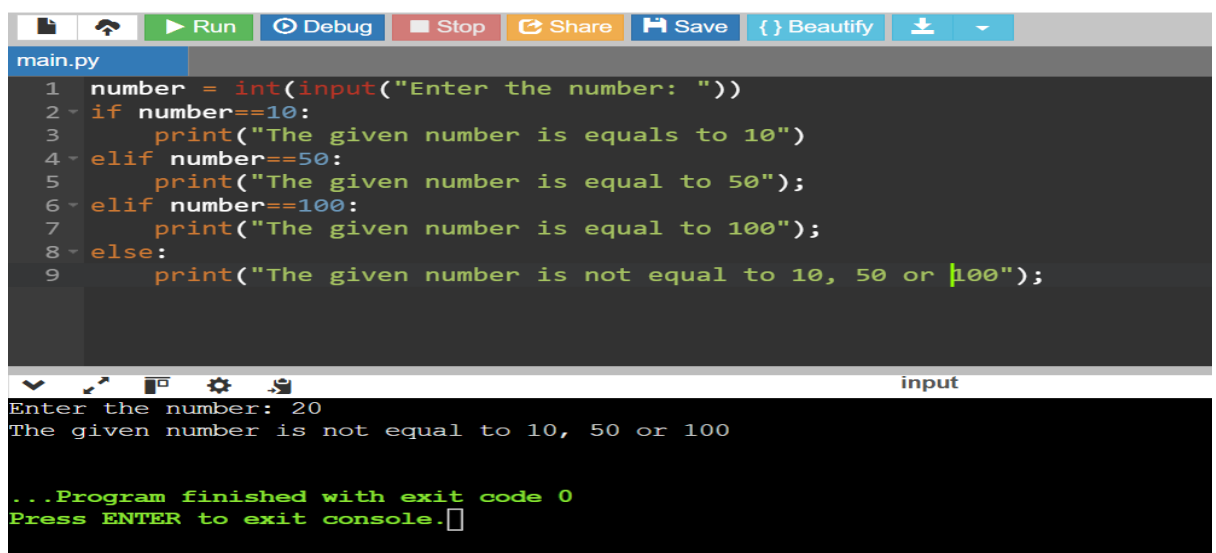
```
main.py
1 age = int (input("Enter your age: "))
2 if age>=18:
3     print("You are eligible to vote !!");
4 else:
5     print("Sorry! you have to wait !!");
```

input

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

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

Python Elif Statement:



```
main.py
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");
```

input

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.

5. Python Loops:

5.1 For loop:

1. Iterating by using index of sequence

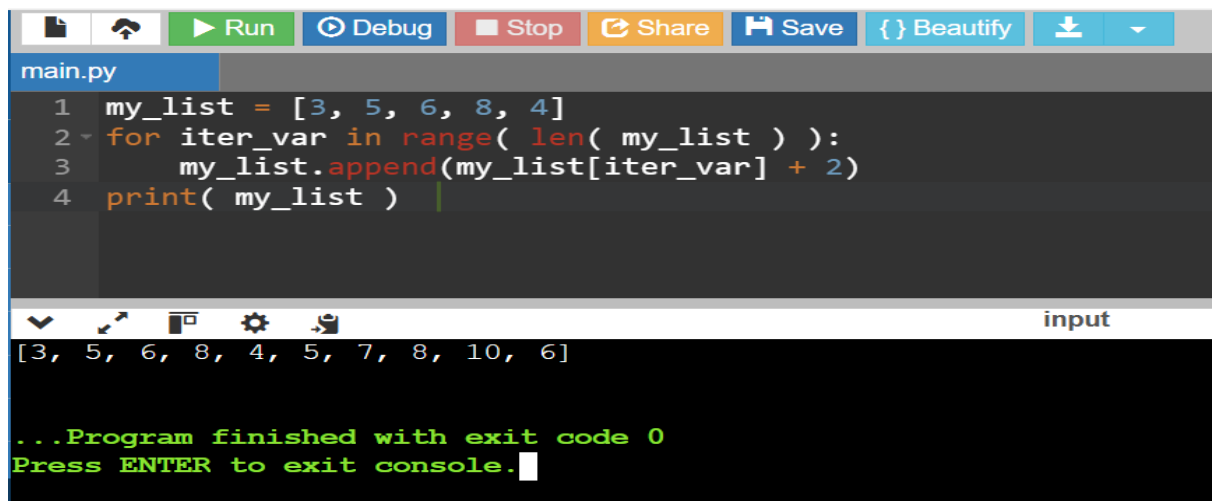


```
main.py
1 numbers = [3, 5, 23, 6, 5, 1, 2, 9, 8]
2 sum_ = 0
3 for num in numbers:
4     sum_ = sum_ + num ** 2
5 print("The sum of squares is: ", sum_)

input
The sum of squares is: 774

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

2. Using Range ()

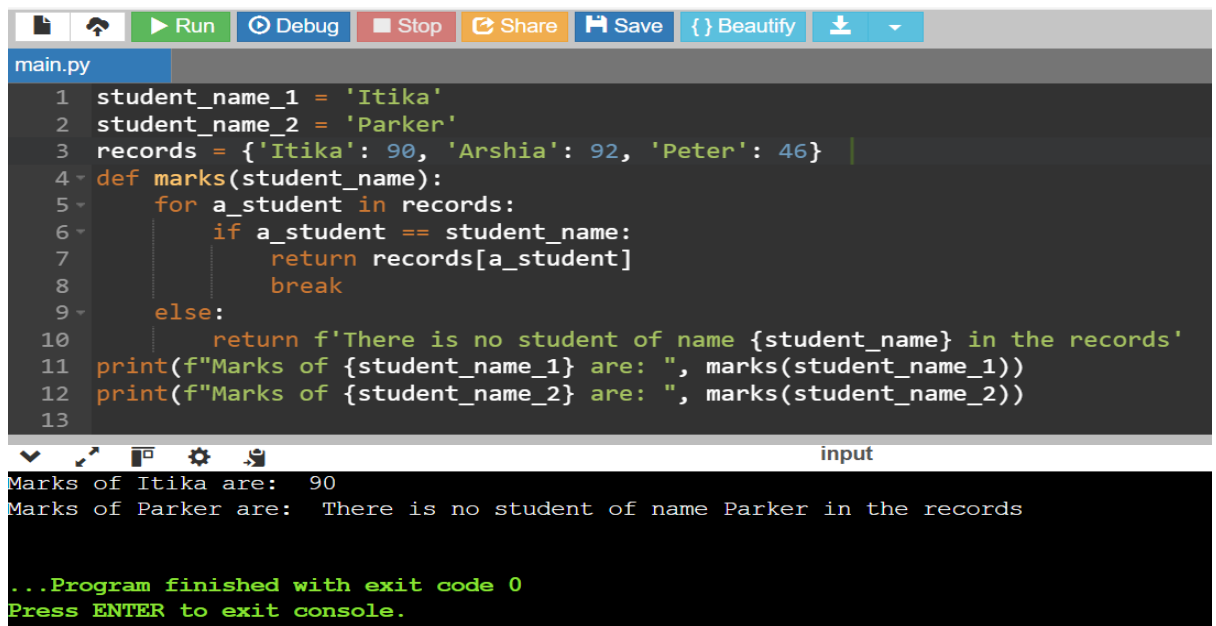


```
main.py
1 my_list = [3, 5, 6, 8, 4]
2 for iter_var in range( len( my_list ) ):
3     my_list.append(my_list[iter_var] + 2)
4 print( my_list )

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 For 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 For 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 = " " )
```

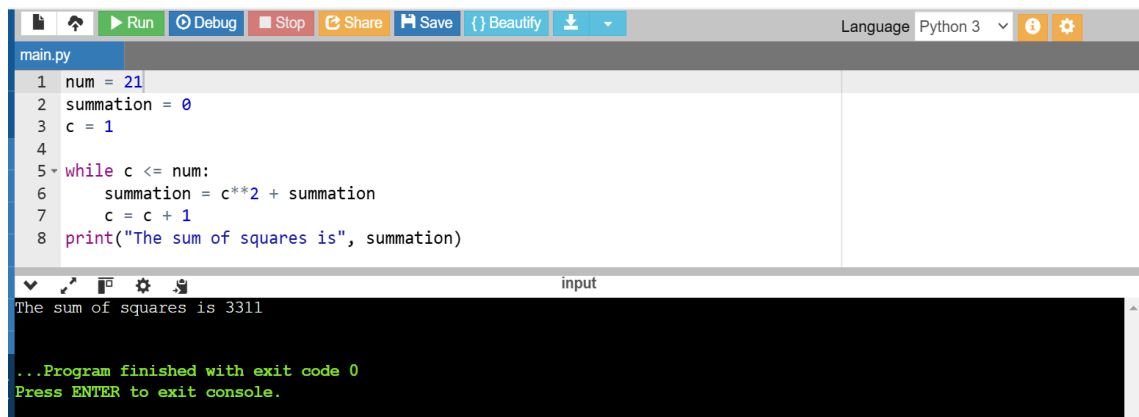
input

0 1 2 3 4 5 6 7 9 10

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

5.2 While loop:

1. Sum of squares



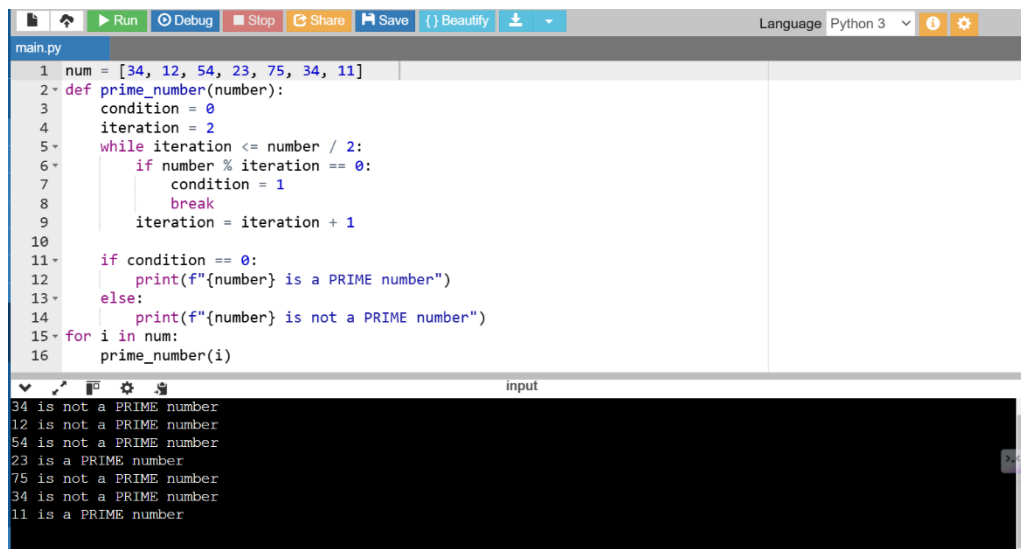
```
main.py
1 num = 21
2 summation = 0
3 c = 1
4
5 while c <= num:
6     summation = c**2 + summation
7     c = c + 1
8 print("The sum of squares is", summation)
```

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

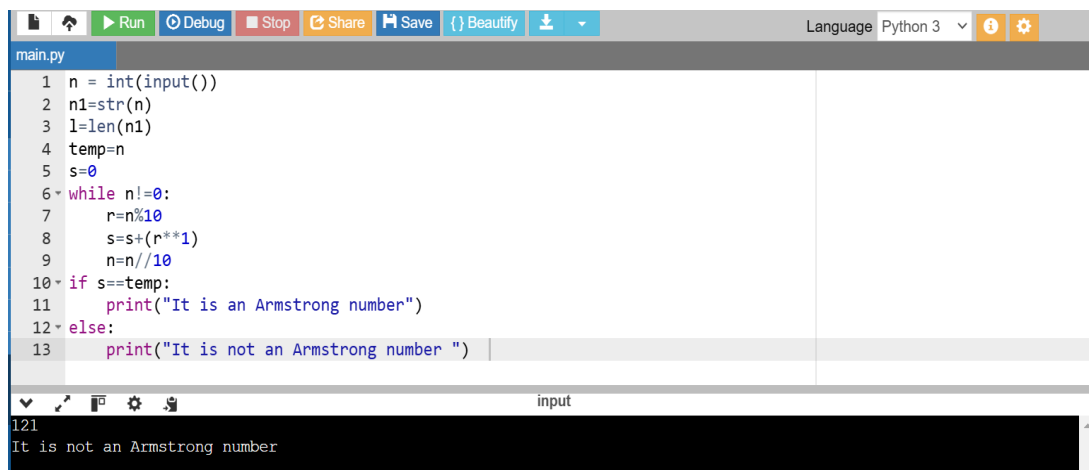


```
main.py
1 num = [34, 12, 54, 23, 75, 34, 11]
2 def prime_number(number):
3     condition = 0
4     iteration = 2
5     while iteration <= number / 2:
6         if number % iteration == 0:
7             condition = 1
8             break
9         iteration = iteration + 1
10
11     if condition == 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

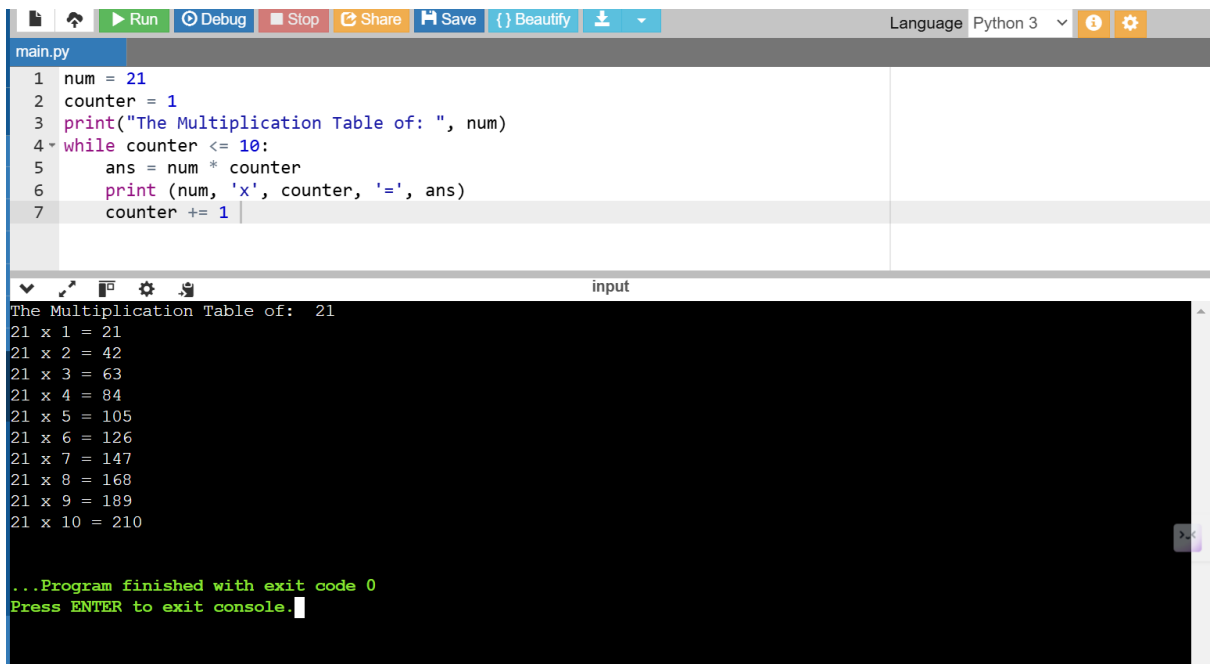


```
main.py
1 n = int(input())
2 n1=str(n)
3 l=len(n1)
4 temp=n
5 s=0
6 while n!=0:
7     r=n%10
8     s=s+(r**l)
9     n=n//10
10 if s==temp:
11     print("It is an Armstrong number")
12 else:
13     print("It is not an Armstrong number ")
```

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 is as follows:

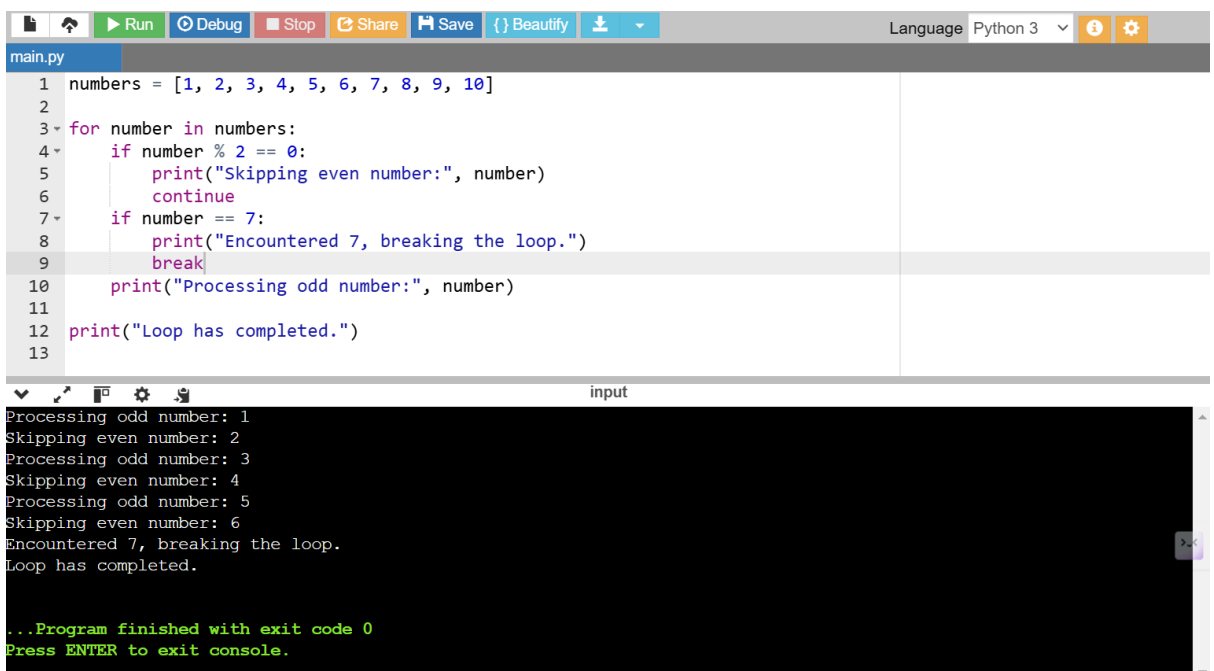
```
1 num = 21
2 counter = 1
3 print("The Multiplication Table of: ", num)
4 while counter <= 10:
5     ans = num * counter
6     print (num, 'x', counter, '=', ans)
7     counter += 1
```

The output in the console is:

```
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.
```

5. BREAK Statement



The screenshot shows a Python IDE with a file named 'main.py'. The code is as follows:

```
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
```

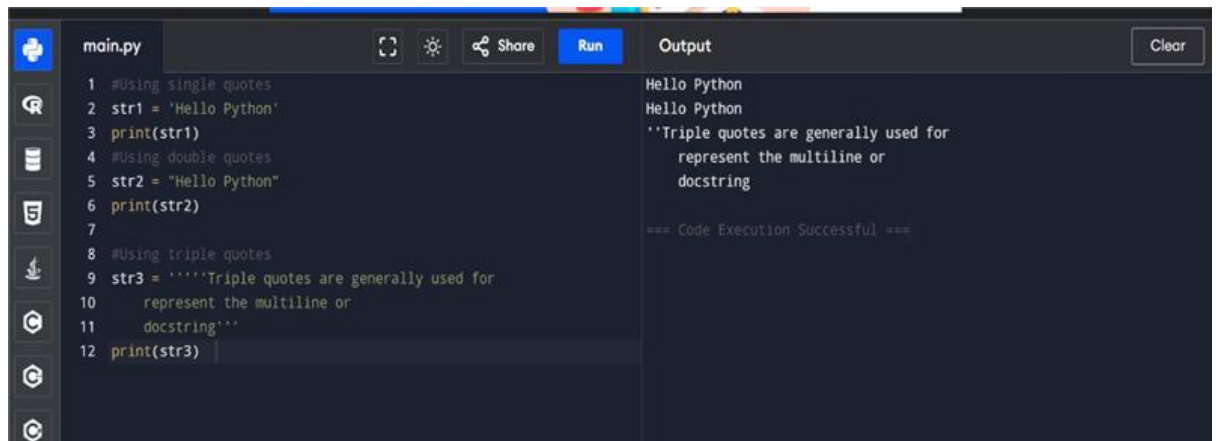
The output in the console is:

```
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.
```

6. PYTHON STRINGS

1. Creating String in Python



The screenshot shows a Python IDE with a file named `main.py`. The code defines three strings: `str1` (single quotes), `str2` (double quotes), and `str3` (triple quotes). The output shows the printed values of these strings, with `str3` demonstrating multiline and docstring capabilities. The code execution is successful.

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

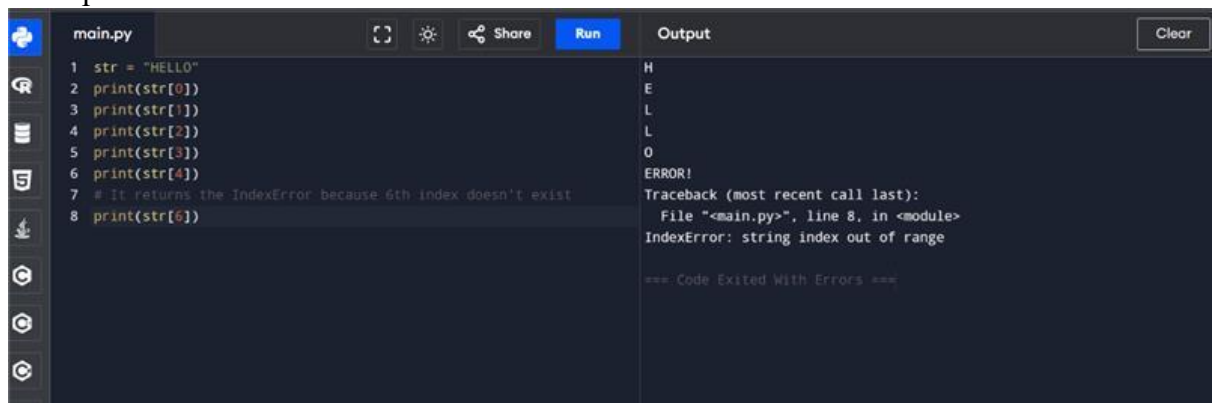
Output:

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

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

2. Strings indexing and splitting

Example 1:



The screenshot shows a Python IDE with a file named `main.py`. The code attempts to access characters of the string "HELLO" using indices 0 through 4, then tries to access index 6, which results in an `IndexError`. The output shows the characters 'H', 'E', 'L', 'L', 'O' and the error message.

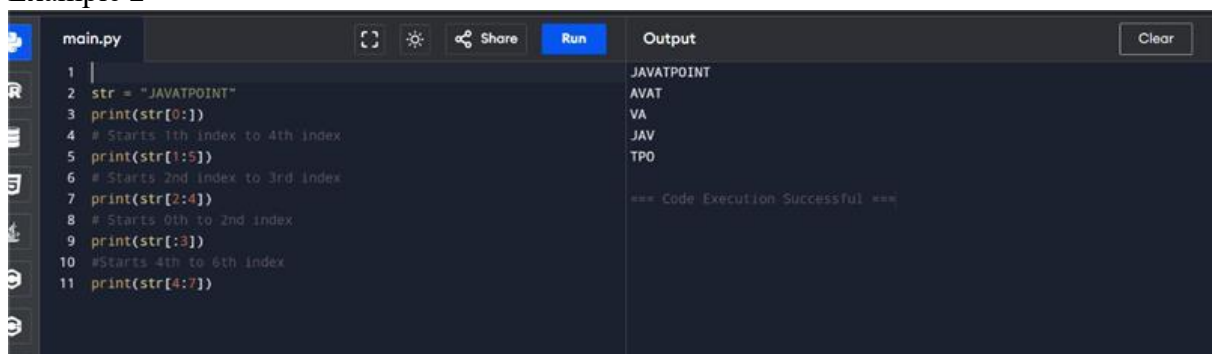
```
1 str = "HELLO"
2 print(str[0])
3 print(str[1])
4 print(str[2])
5 print(str[3])
6 print(str[4])
7 # It returns the IndexError because 6th index doesn't exist
8 print(str[6])
```

Output:

```
H
E
L
L
O
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 8, in <module>
IndexError: string index out of range

=== Code Exited With Errors ===
```

Example 2



The screenshot shows a Python IDE with a file named `main.py`. The code demonstrates various slicing operations on the string "JAVATPOINT". The output shows the results of these slices: the full string, and substrings from different starting and ending indices.

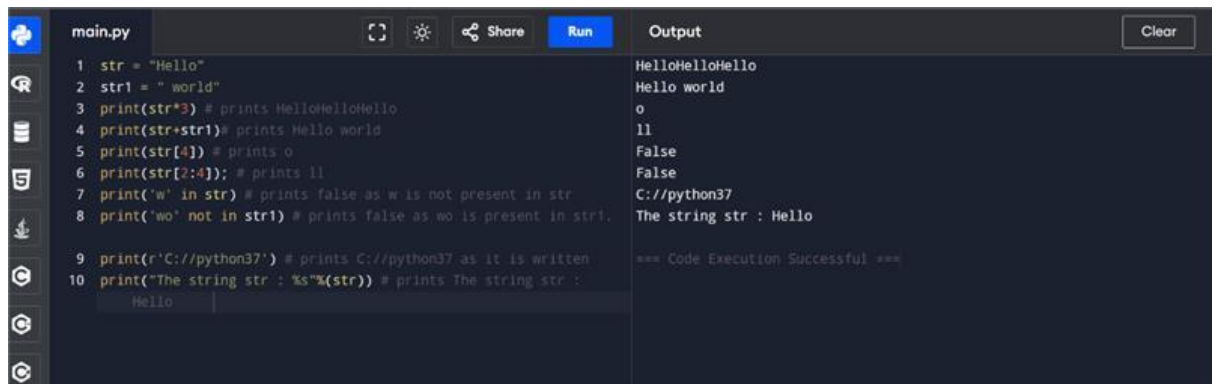
```
1
2 str = "JAVATPOINT"
3 print(str[0:])
4 # Starts 1th index to 4th index
5 print(str[1:5])
6 # Starts 2nd index to 3rd index
7 print(str[2:4])
8 # Starts 0th to 2nd index
9 print(str[:3])
10 #Starts 4th to 6th index
11 print(str[4:7])
```

Output:

```
JAVATPOINT
AVAT
VA
JAV
TPO

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

3. Strings Operators



The screenshot shows a Python IDE with a file named 'main.py'. The code in the editor demonstrates various string operations: concatenation, indexing, slicing, membership testing, and string formatting. The output window on the right shows the results of these operations, including 'HelloHelloHello', 'Hello world', 'o', 'll', 'False', 'False', 'C://python37', and 'The string str : Hello'. The code execution is successful.

```
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.

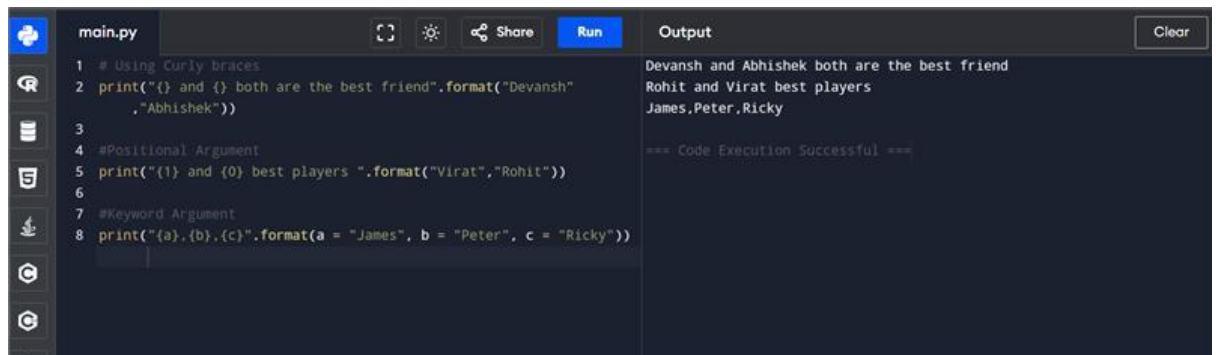
9 print(r'C://python37') # prints C://python37 as it is written
10 print("The string str : %s"%(str)) # prints The string str :
    Hello
```

Output:

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

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

4. String formatting



The screenshot shows a Python IDE with a file named 'main.py'. The code demonstrates three different ways to format strings: using curly braces, positional arguments, and keyword arguments. The output window shows the formatted strings: 'Devansh and Abhishek both are the best friend', 'Rohit and Virat best players', and 'James,Peter,Ricky'. The code execution is successful.

```
1 # Using Curly braces
2 print("{} and {} both are the best friend".format("Devansh"
    ,"Abhishek"))
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 = "James", b = "Peter", c = "Ricky"))
```

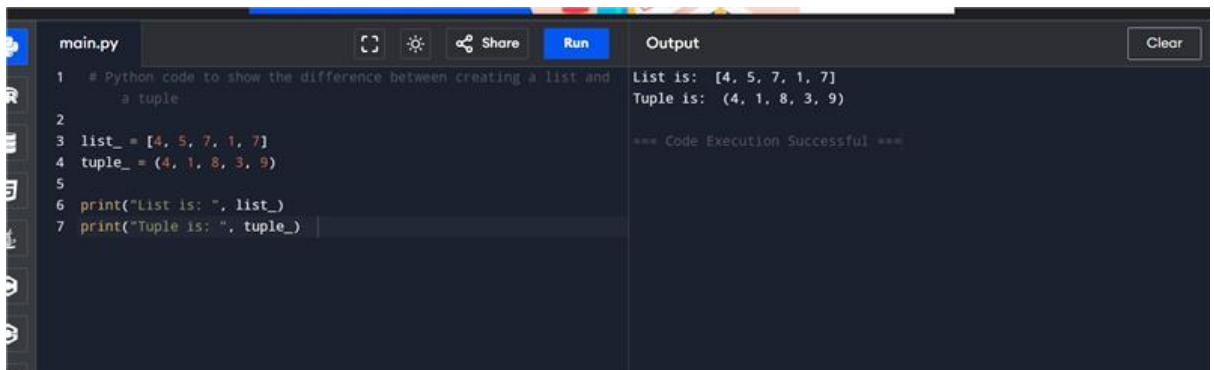
Output:

```
Devansh and Abhishek both are the best friend
Rohit and Virat best players
James,Peter,Ricky

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

7. Lists and Tuples

1. List and Tuple Syntax Differences



The screenshot shows a Python IDE with a file named `main.py`. The code defines a list `list_` and a tuple `tuple_`. The list is created with square brackets and the tuple with parentheses. The output window shows the printed values of both variables.

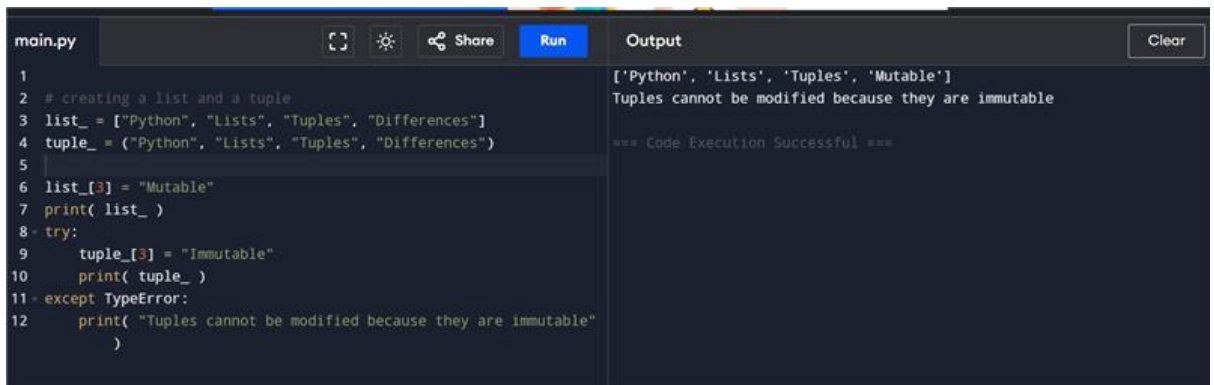
```
1 # Python code to show the difference between creating a list and a tuple
2
3 list_ = [4, 5, 7, 1, 7]
4 tuple_ = (4, 1, 8, 3, 9)
5
6 print("List is: ", list_)
7 print("Tuple is: ", tuple_)
```

Output:

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

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

2. Updating the element of list and tuple at a particular index



The screenshot shows a Python IDE with a file named `main.py`. The code creates a list `list_` and a tuple `tuple_`. It attempts to update the element at index 3 of both. The list update is successful, but the tuple update raises a `TypeError` because tuples are immutable. The output window shows the successful list update and the error message for the tuple.

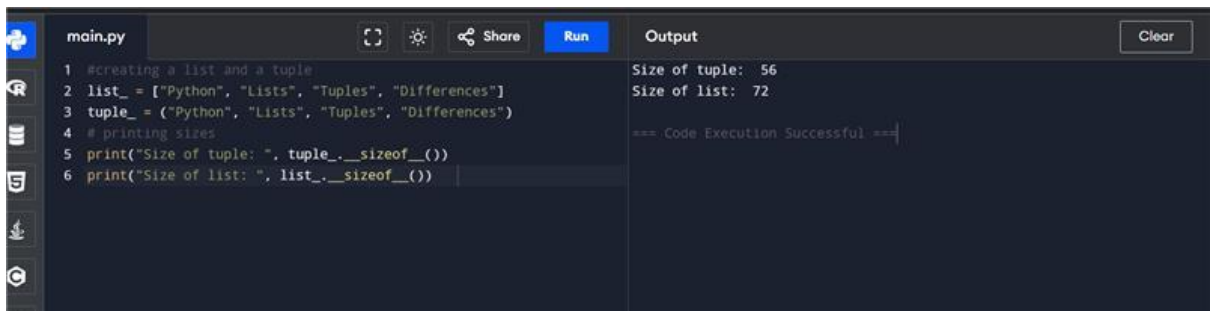
```
1 # creating a list and a tuple
2
3 list_ = ["Python", "Lists", "Tuples", "Differences"]
4 tuple_ = ("Python", "Lists", "Tuples", "Differences")
5
6 list_[3] = "Mutable"
7 print( list_ )
8
9 try:
10     tuple[3] = "Immutable"
11 except TypeError:
12     print( "Tuples cannot be modified because they are immutable" )
```

Output:

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

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

3. Code to show the difference in the size of a list and a tuple



The screenshot shows a Python IDE with a file named `main.py`. The code creates a list `list_` and a tuple `tuple_` and prints their sizes using the `__sizeof__()` method. The output window shows the sizes of both variables.

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

Output:

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

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