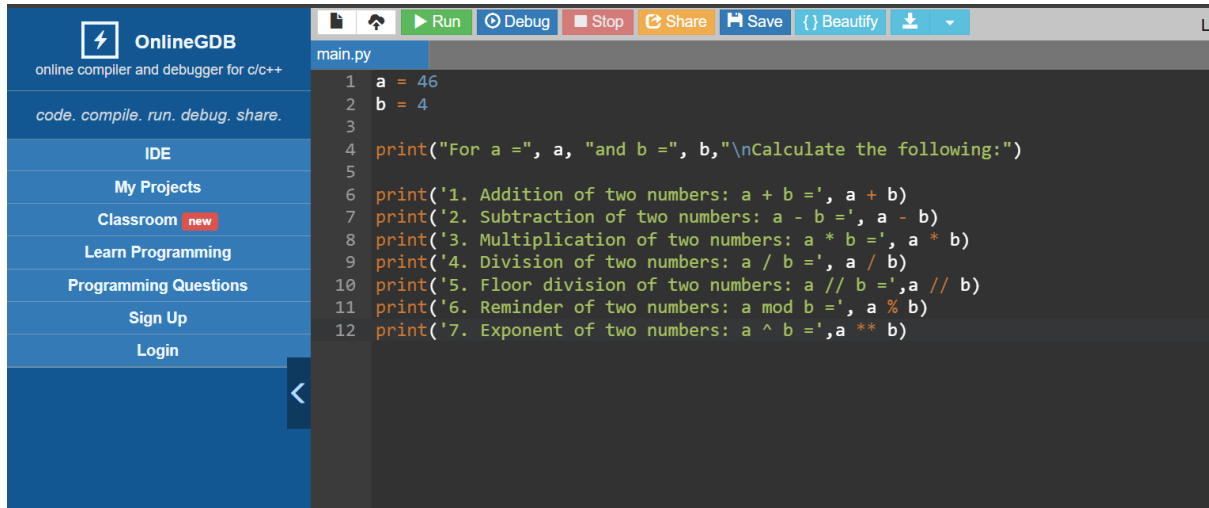


PYTHON OPERATORS

Code:

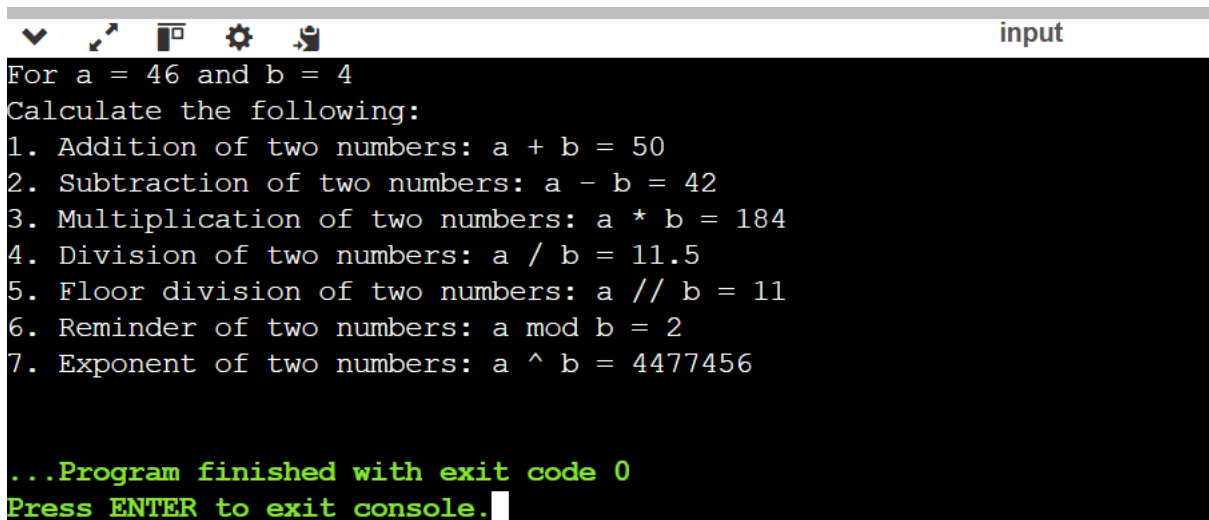
Arithmetic:



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area displays a file named 'main.py' with the following Python code:

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

Output:



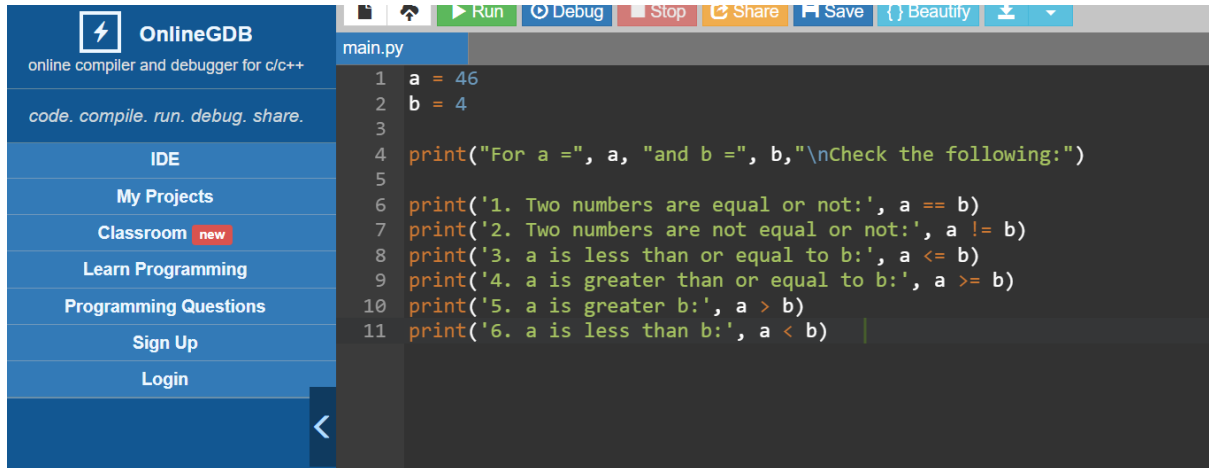
The screenshot shows the output console of the OnlineGDB IDE. The output text is as follows:

```
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.
```

Code:

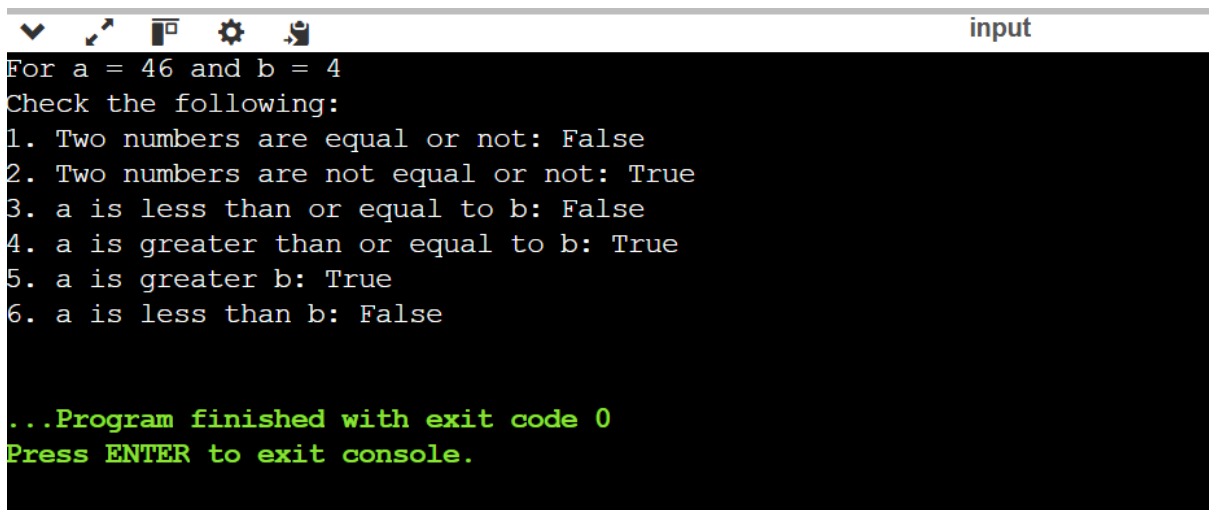
2.Comparison Operators:



The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links: IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area displays a file named 'main.py' with the following Python code:

```
1 a = 46
2 b = 4
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:



The screenshot shows the output console of the OnlineGDB IDE. The output text is as follows:

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

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

Code:

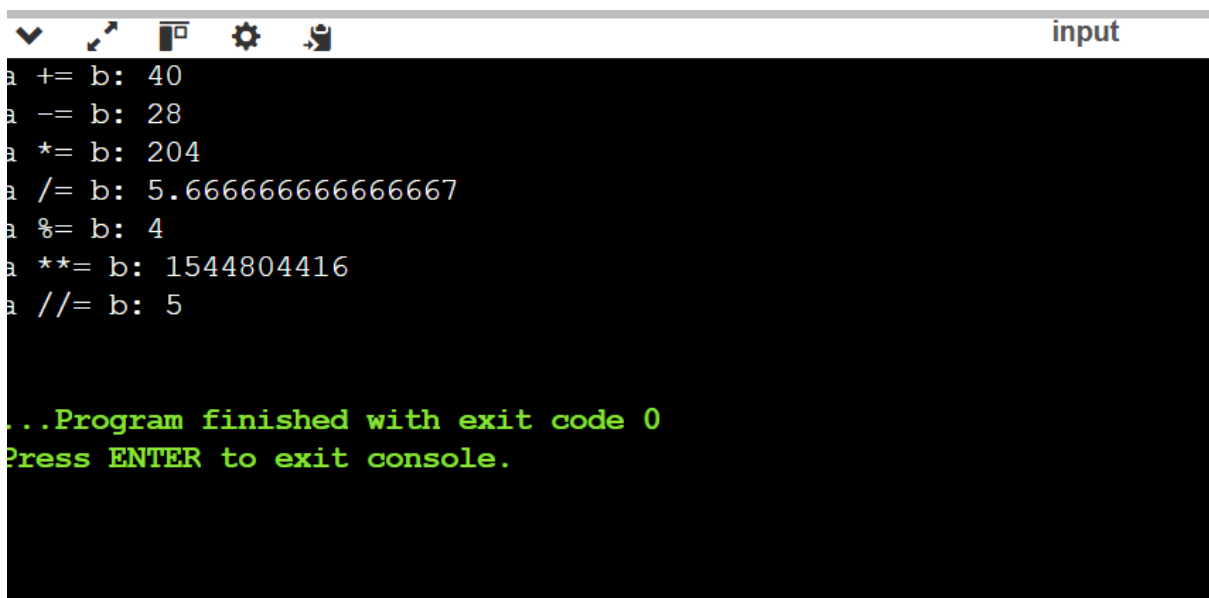
3. Assignment Operators:



The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links: IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Sign Up, and Login. The top of the editor has a toolbar with icons for file operations, a 'Run' button, 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. The main editor area displays a file named 'main.py' with the following Python code:

```
1 a = 34
2 b = 6
3
4
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)
10 print('a **= b:', a ** b)
11 print('a //= b:', a // b)
```

Output:



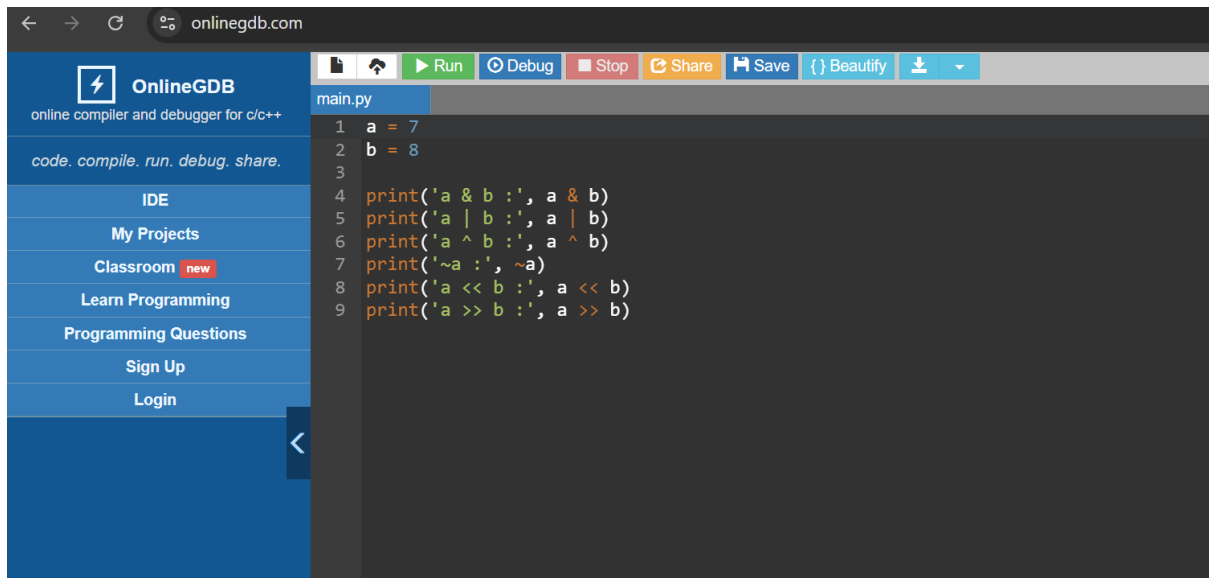
The screenshot shows the output console of the OnlineGDB IDE. The output displays the results of the Python script's print statements, showing the values of 'a' after each operation. The output is as follows:

```
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.
```

Code:

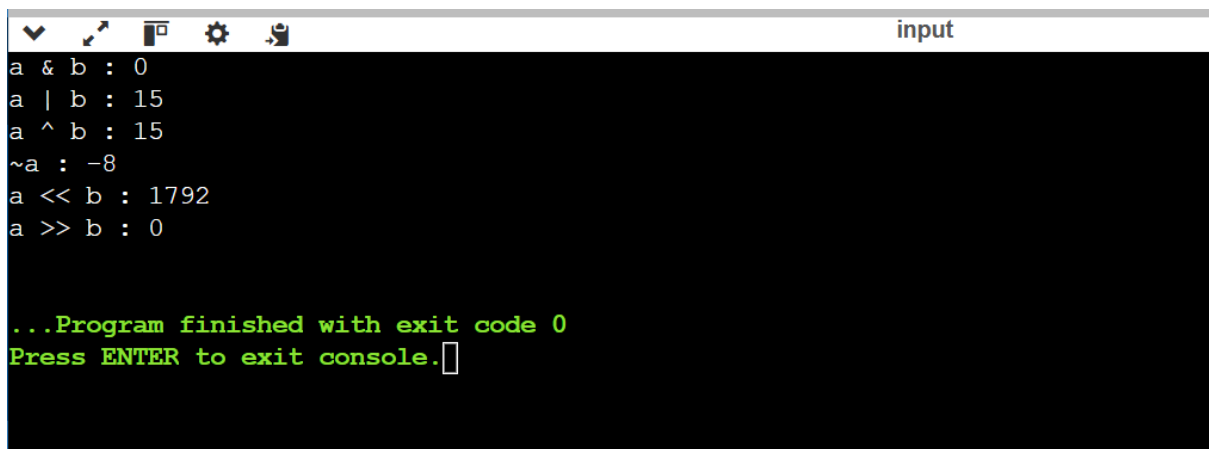
4.Bitwise Operators:



The screenshot shows the OnlineGDB web interface. The left sidebar contains navigation links: IDE, My Projects, Classroom (marked as new), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area displays a file named 'main.py' with the following Python code:

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

Output:



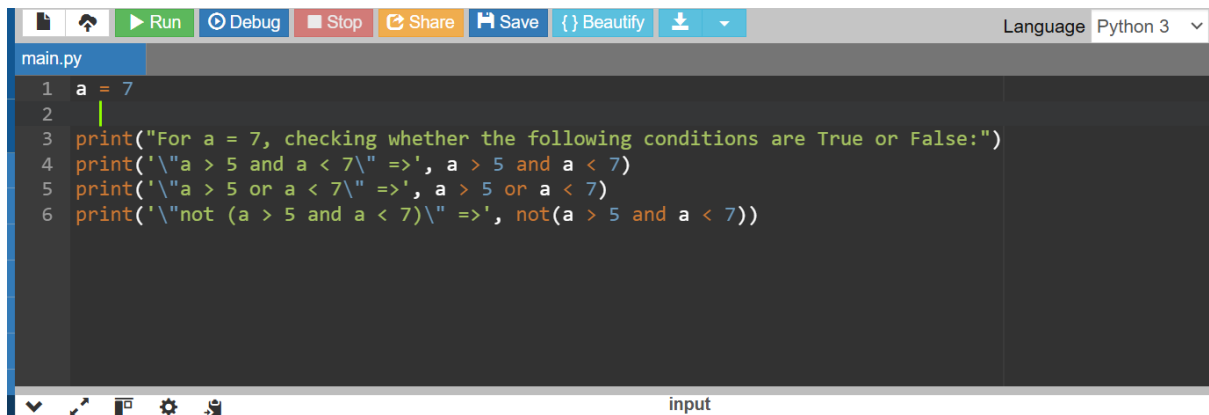
The screenshot shows the console output of the program. The output is as follows:

```
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.
```

Code:

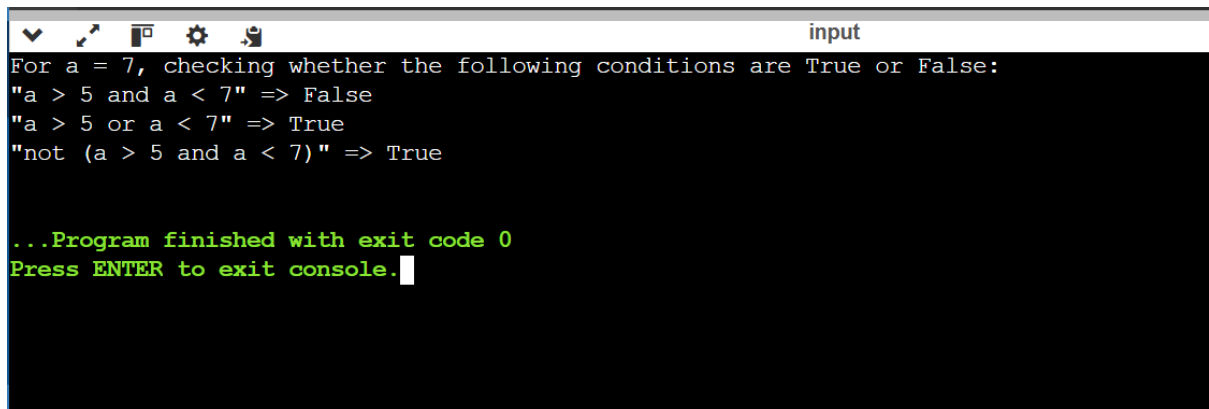
5.Logical Operators:



The screenshot shows a code editor window with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to Python 3. The file is named main.py. The code is as follows:

```
1 a = 7
2
3 print("For a = 7, checking whether the following conditions are True or False:")
4 print('\na > 5 and a < 7" =>', a > 5 and a < 7)
5 print('\na > 5 or a < 7" =>', a > 5 or a < 7)
6 print('\nnot (a > 5 and a < 7)" =>', not(a > 5 and a < 7))
```

Output:



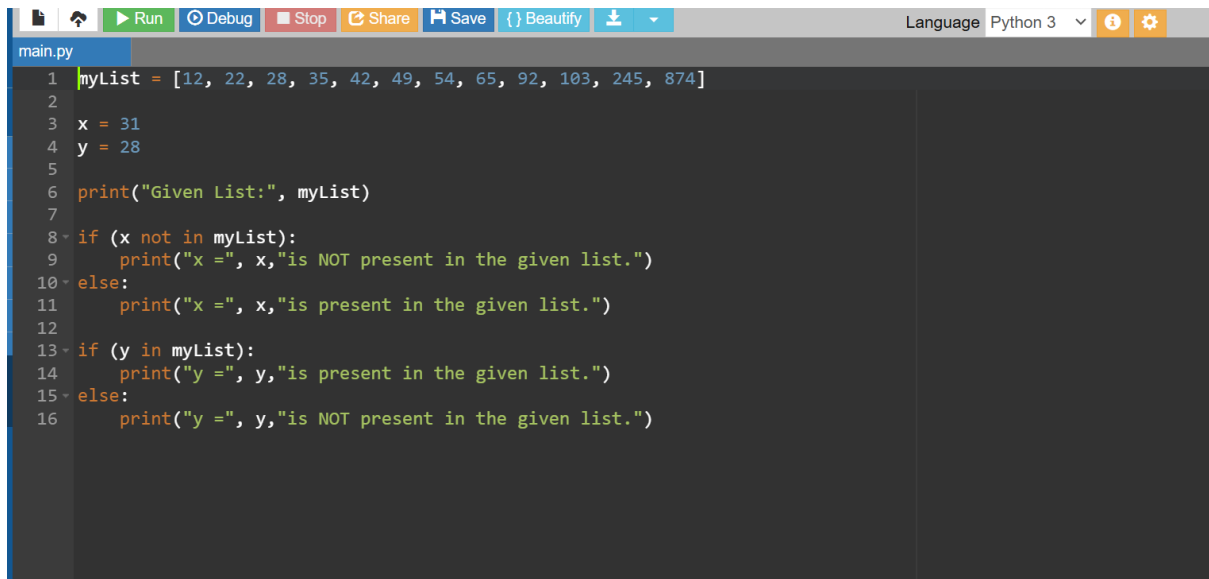
The screenshot shows a terminal window with the following output:

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

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

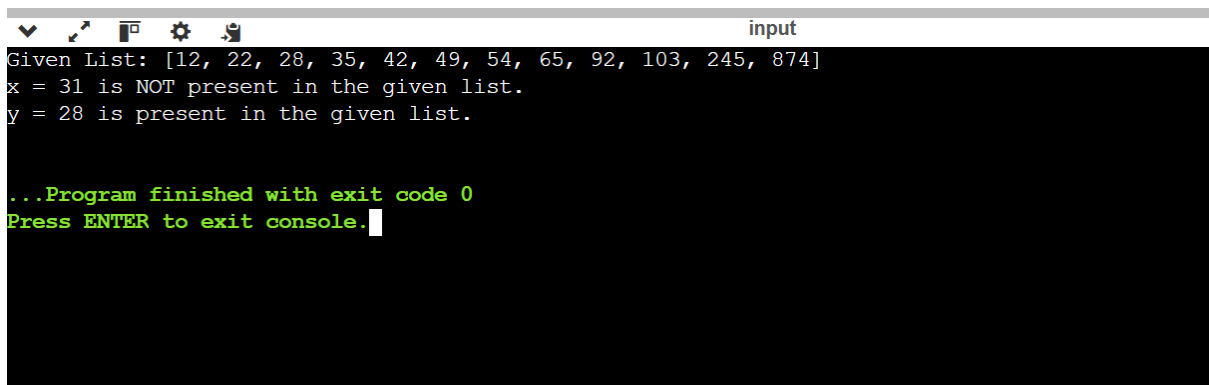
Code:

6.Membership operators:



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

OUTPUT:

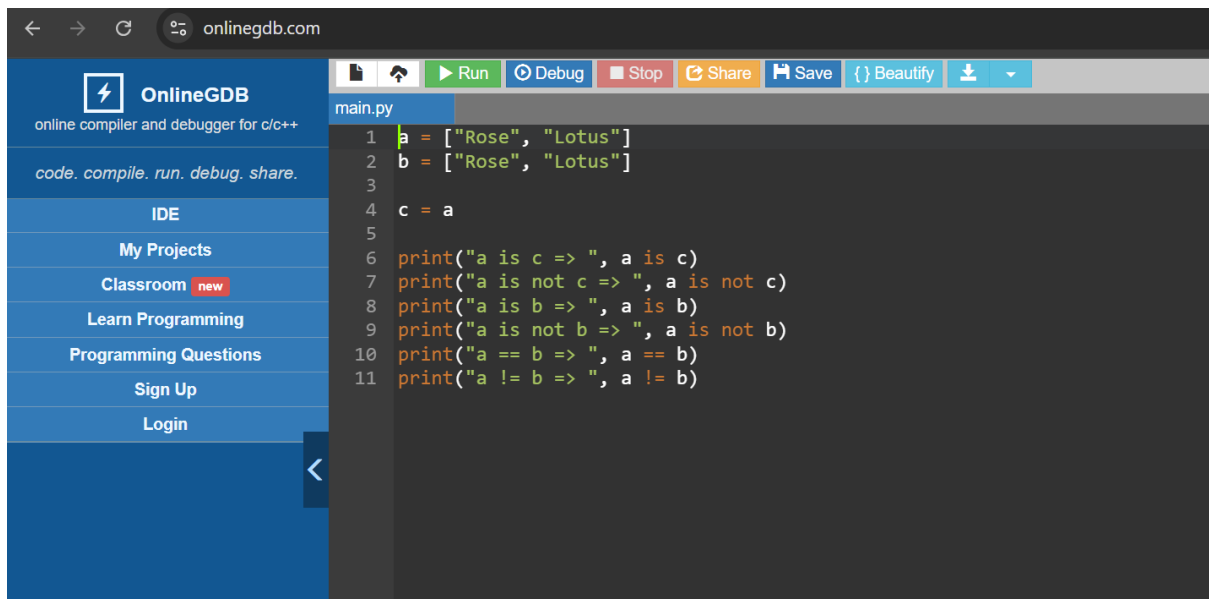


```
input
Given List: [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]
x = 31 is NOT present in the given list.
y = 28 is present in the given list.

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

Code:

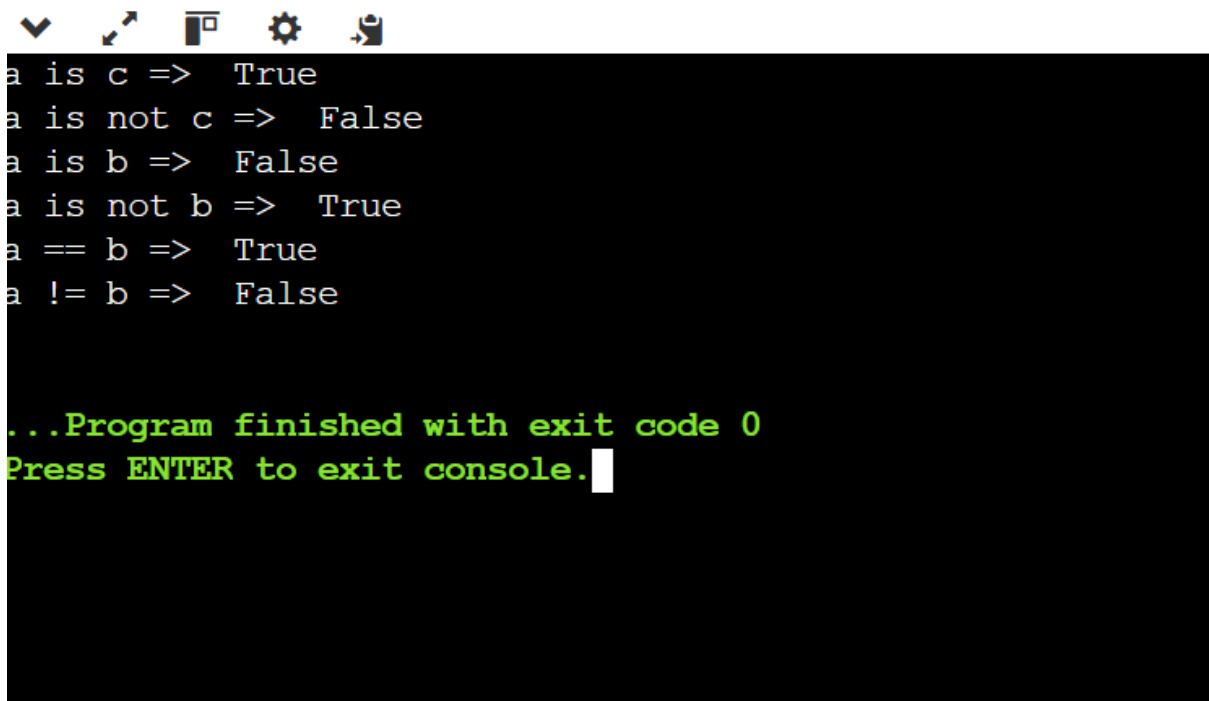
7.Identity Operators:



The screenshot shows the OnlineGDB web interface. The browser address bar displays 'onlinegdb.com'. The interface includes a sidebar with navigation links: 'IDE', 'My Projects', 'Classroom' (with a 'new' badge), 'Learn Programming', 'Programming Questions', 'Sign Up', and 'Login'. The main editor area shows a file named 'main.py' with the following Python code:

```
1 a = ["Rose", "Lotus"]
2 b = ["Rose", "Lotus"]
3
4 c = a
5
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)
```

Output:



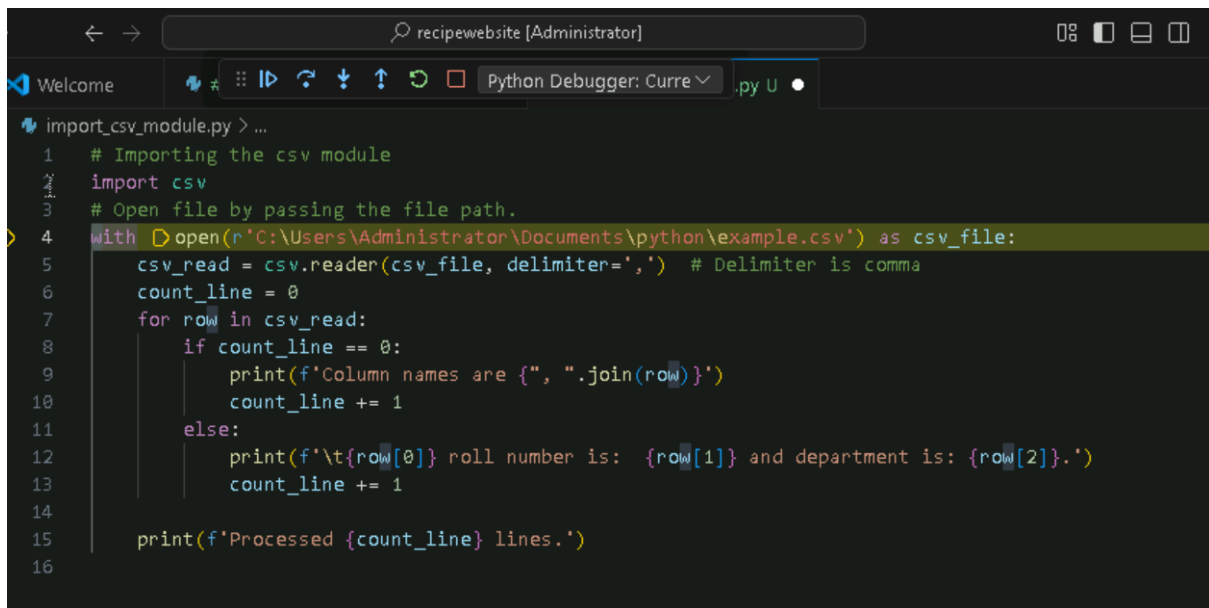
The screenshot shows the output of the Python script. The output is displayed in a console window with a dark background and light green text. The output is as follows:

```
a is c => True
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.
```

READ CSV FILE IN PYTHON:

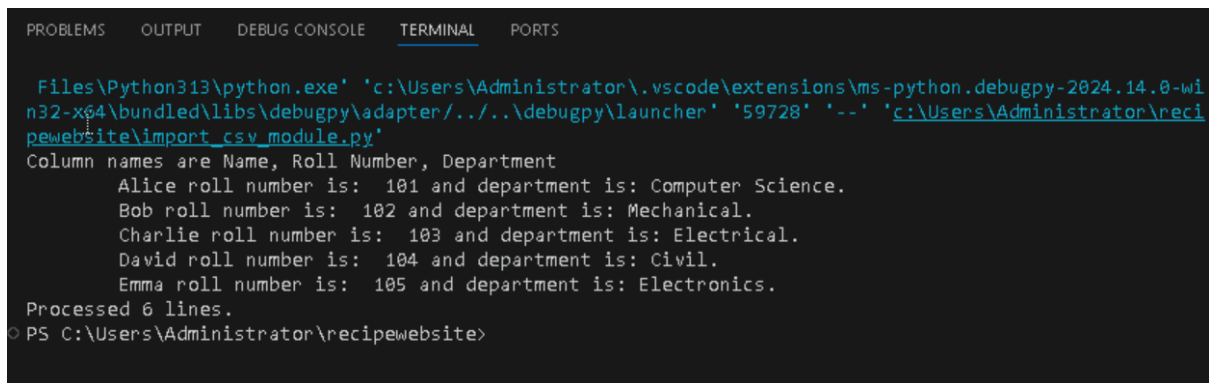
1. CODE:



The screenshot shows a Python script named `import_csv_module.py` being executed in a Python Debugger. The script reads a CSV file located at `r'C:\Users\Administrator\Documents\python\example.csv'`. The CSV file contains 6 lines of data. The script prints the column names and then iterates through each row, printing the roll number and department for each person.

```
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:

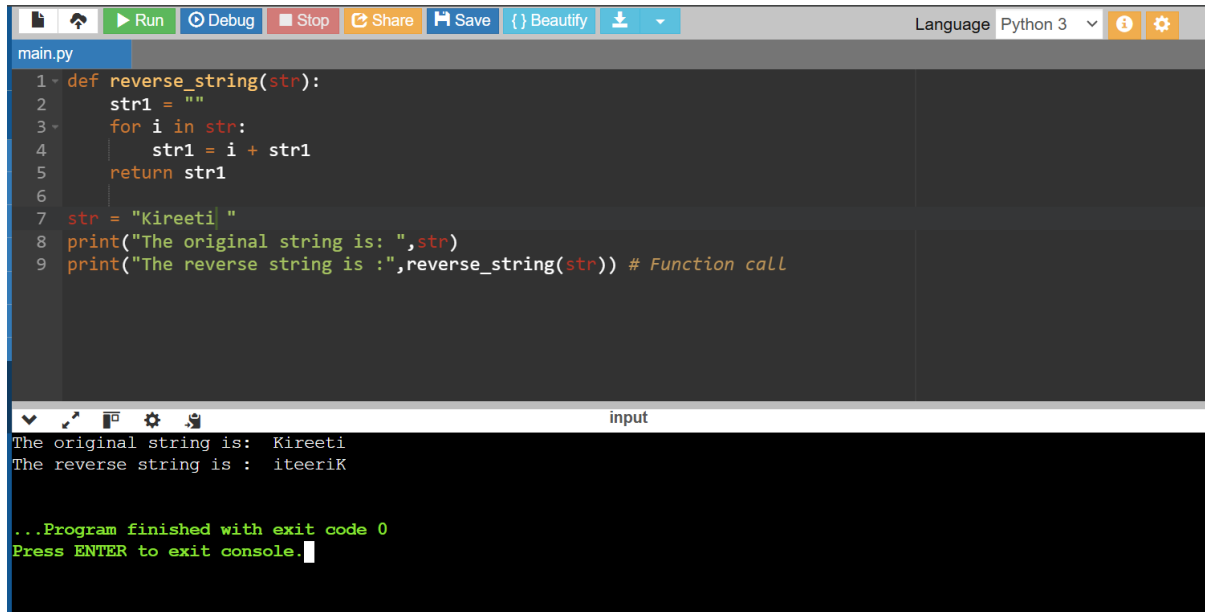


The screenshot shows the terminal output of the Python script. The output displays the column names and then iterates through each row, printing the roll number and department for each person. The script has processed 6 lines.

```
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:



The screenshot shows a Python IDE with a file named 'main.py'. The code defines a function 'reverse_string' that takes a string 'str' and returns its reverse by iterating through each character and building a new string 'str1'. The main code sets 'str' to 'Kireeti', prints the original string, and then prints the reversed string by calling 'reverse_string(str)'. The output in the console shows 'The original string is: Kireeti' and 'The reverse string is : iteerik'. The program finishes with exit code 0.

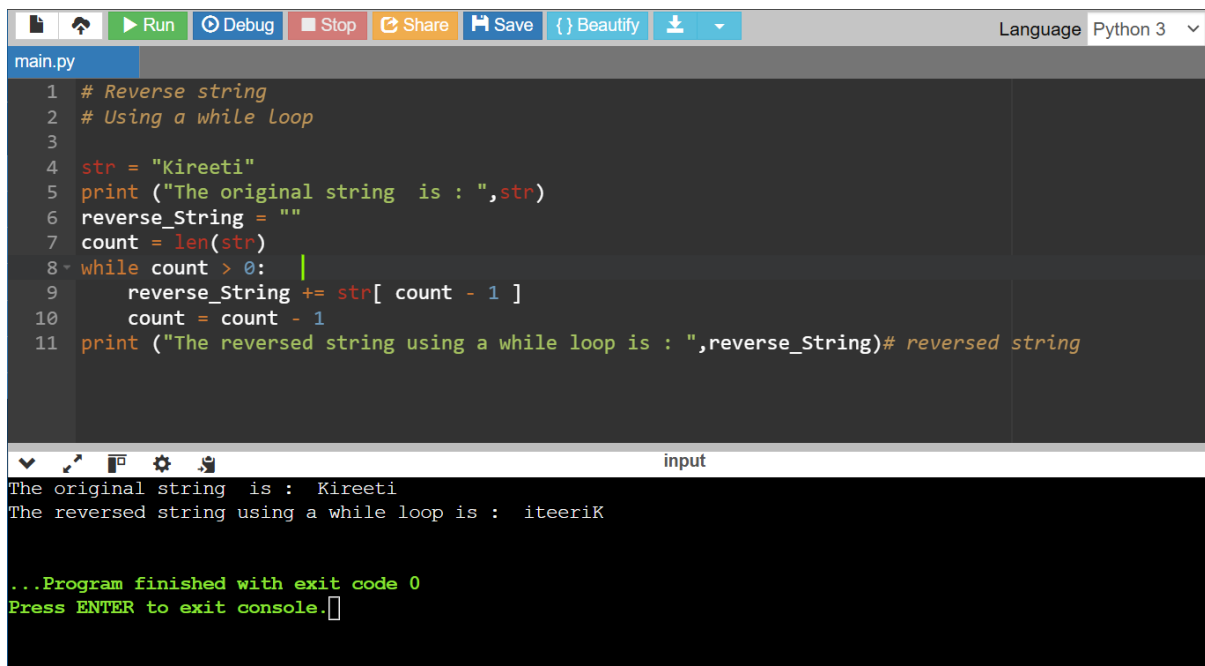
```
1 def reverse_string(str):
2     str1 = ""
3     for i in str:
4         str1 = i + str1
5     return str1
6
7 str = "Kireeti"
8 print("The original string is: ",str)
9 print("The reverse string is :",reverse_string(str)) # Function call
```

input

The original string is: Kireeti
The reverse string is : iteerik

...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 defines a function 'reverse_string' that takes a string 'str' and returns its reverse by iterating through each character and building a new string 'str1'. The main code sets 'str' to 'Kireeti', prints the original string, and then prints the reversed string by calling 'reverse_string(str)'. The output in the console shows 'The original string is: Kireeti' and 'The reverse string is : iteerik'. The program finishes with exit code 0.

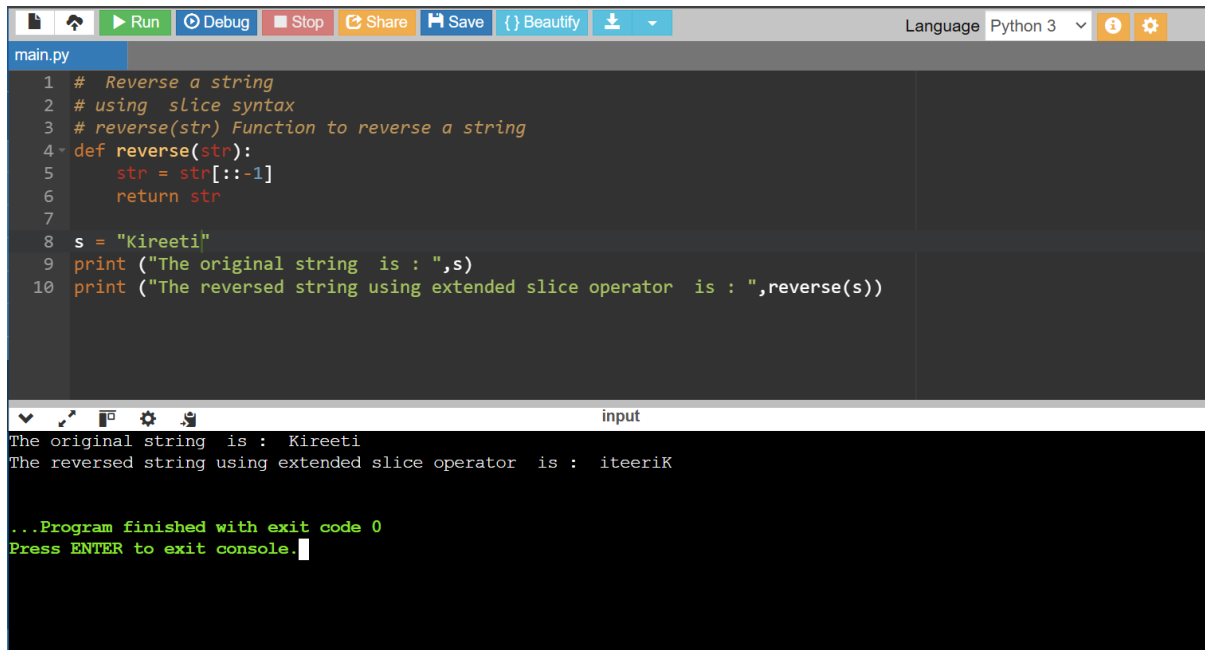
```
1 # Reverse string
2 # Using a while loop
3
4 str = "Kireeti"
5 print ("The original string is : ",str)
6 reverse_String = ""
7 count = len(str)
8 while count > 0:
9     reverse_String += str[ count - 1 ]
10    count = count - 1
11 print ("The reversed string using a while loop is : ",reverse_String)#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

input

The original string is : Kireeti
The reversed string using a while loop is : iteerik

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

3.USING SLICE OPERATOR:



The screenshot shows a Python IDE with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to Python 3. The editor displays a file named 'main.py' with the following code:

```
1 # Reverse a string
2 # using slice syntax
3 # reverse(str) Function to reverse a string
4 def reverse(str):
5     str = str[::-1]
6     return str
7
8 s = "Kireeti"
9 print ("The original string is : ",s)
10 print ("The reversed string using extended slice operator is : ",reverse(s))
```

The console output at the bottom shows the program's execution:

```
input
The original string is : Kireeti
The reversed string using extended slice operator is : iteerik

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

4.USING REVERSED FUNCTION INSIDE JOIN:



The screenshot shows a Python IDE with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to Python 3. The editor displays a file named 'main.py' with the following code:

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

The console output at the bottom shows the program's execution:

```
input
The original string is : Kireeti
The reversed string using reversed() is : iteerik

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

5.USING RECURSION:



The image shows a screenshot of a Python IDE interface. The top toolbar includes buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The language is set to Python 3. The editor window, titled 'main.py', contains the following Python code:

```
1 def reverse(str):  
2     if len(str) == 0: # Checking the Lenght of string  
3         return str  
4     else:  
5         return reverse(str[1:]) + str[0]  
6  
7 str = "Kireeti"  
8 print ("The original string is : ", str)  
9 print ("The reversed string(using recursion) is : ", reverse(str))
```

Below the editor is a console window titled 'input'. It displays the output of the program:

```
The original string is : Kireeti  
The reversed string(using recursion) is : iteerik  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```