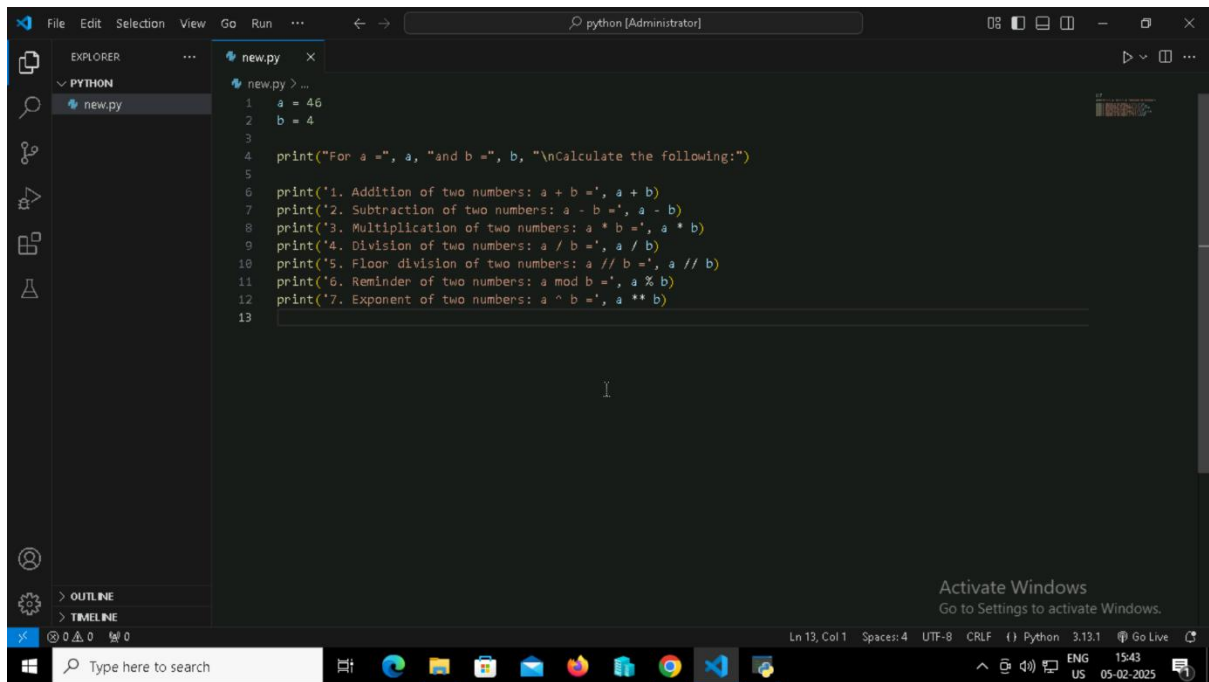


PYTHON OPERATORS

The **Operators** are the symbols used to perform a specific operation on different values and variables.

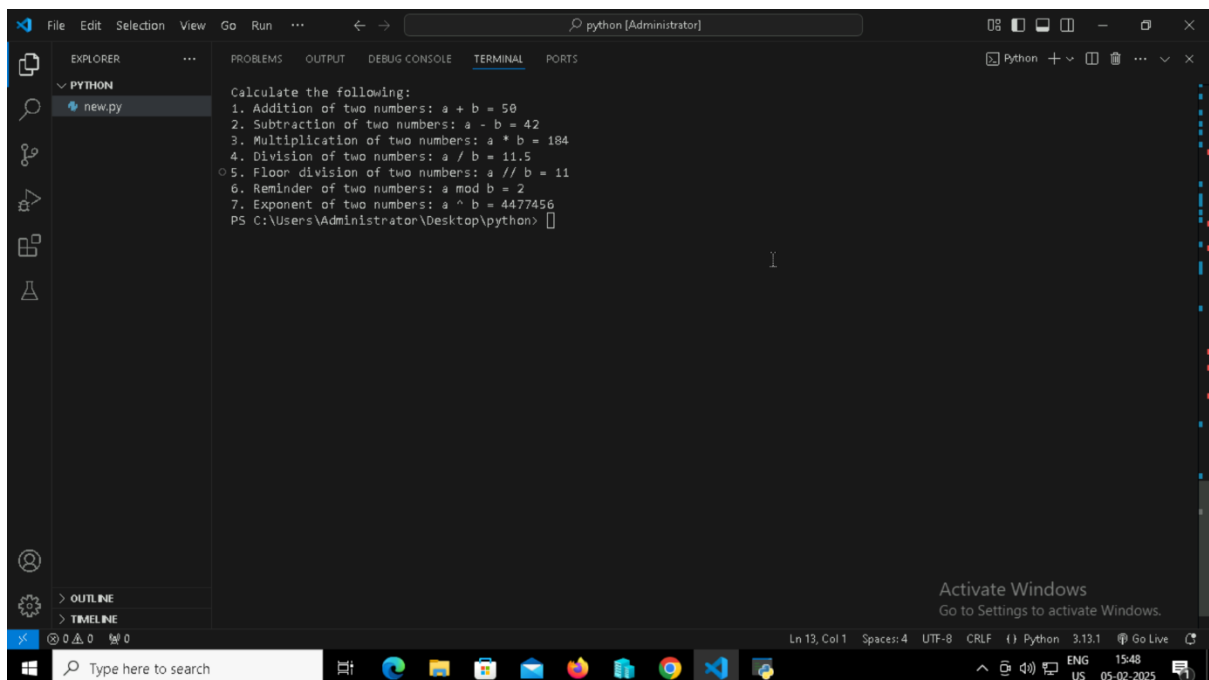
1. These code examples of arithmetic operators in Python:



```
File Edit Selection View Go Run ... python [Administrator]
EXPLORER
PYTHON
new.py
new.py > ...
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. Remainder of two numbers: a mod b =', a % b)
12 print('7. Exponent of two numbers: a ^ b =', a ** b)
13
```

Activate Windows
Go to Settings to activate Windows.

Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.1 Go Live

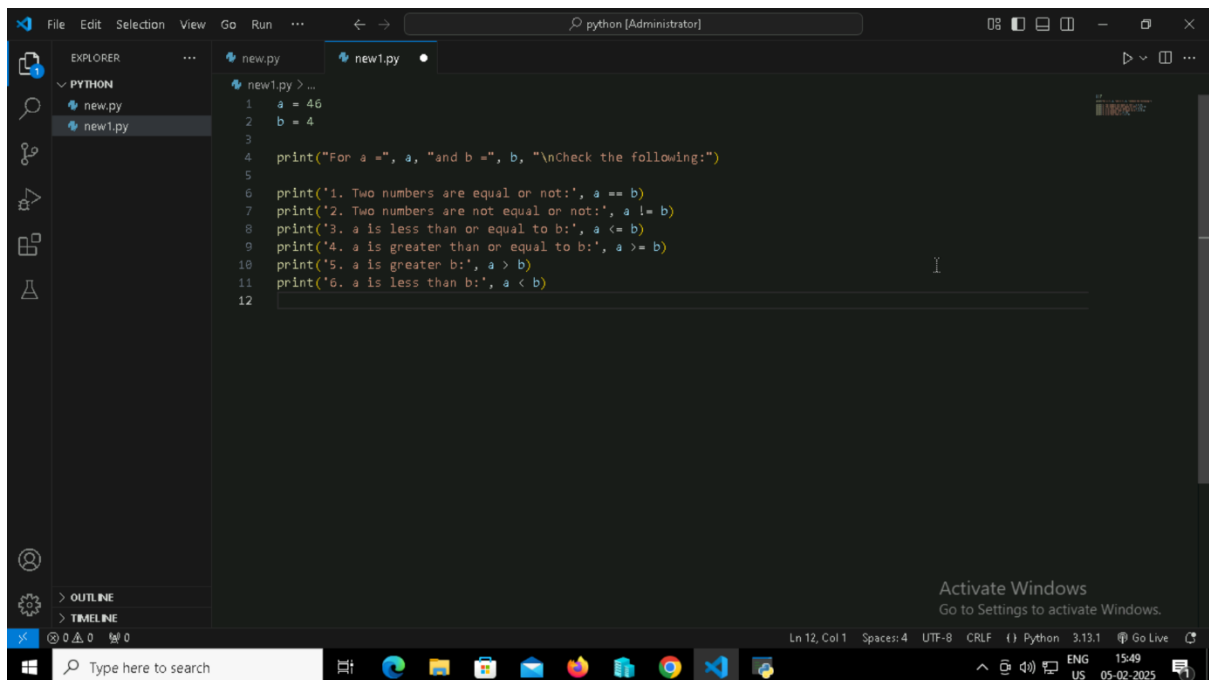


```
File Edit Selection View Go Run ... python [Administrator]
EXPLORER
PYTHON
new.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Calculate the following:
1. Addition of two numbers: a + b = 50
2. Subtraction of two numbers: a - b = 42
3. Multiplication of two numbers: a * b = 184
4. Division of two numbers: a / b = 11.5
5. Floor division of two numbers: a // b = 11
6. Remainder of two numbers: a mod b = 2
7. Exponent of two numbers: a ^ b = 4477456
PS C:\Users\Administrator\Desktop\python>
```

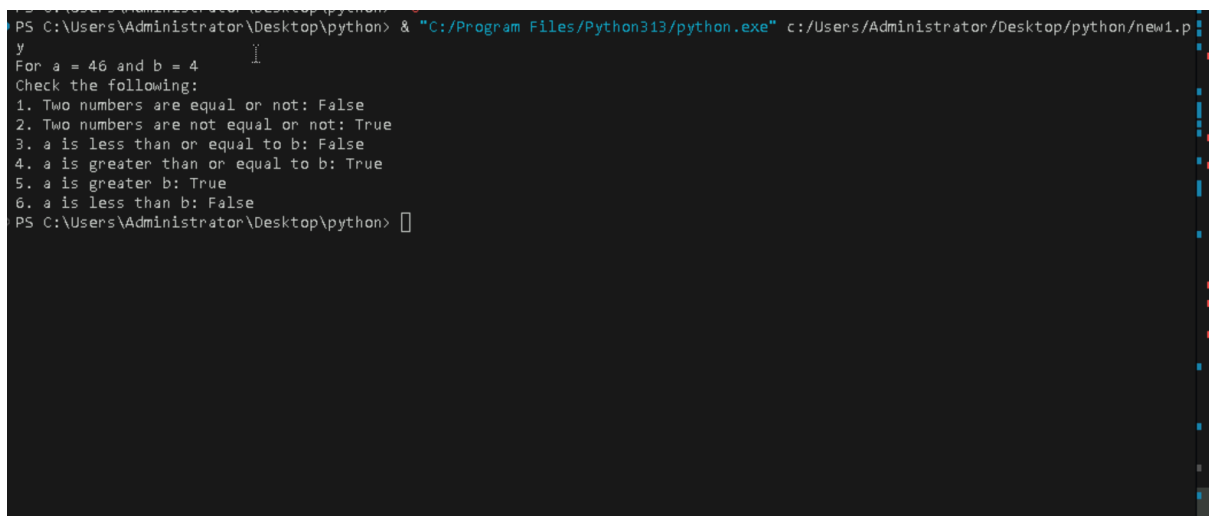
Activate Windows
Go to Settings to activate Windows.

Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.1 Go Live

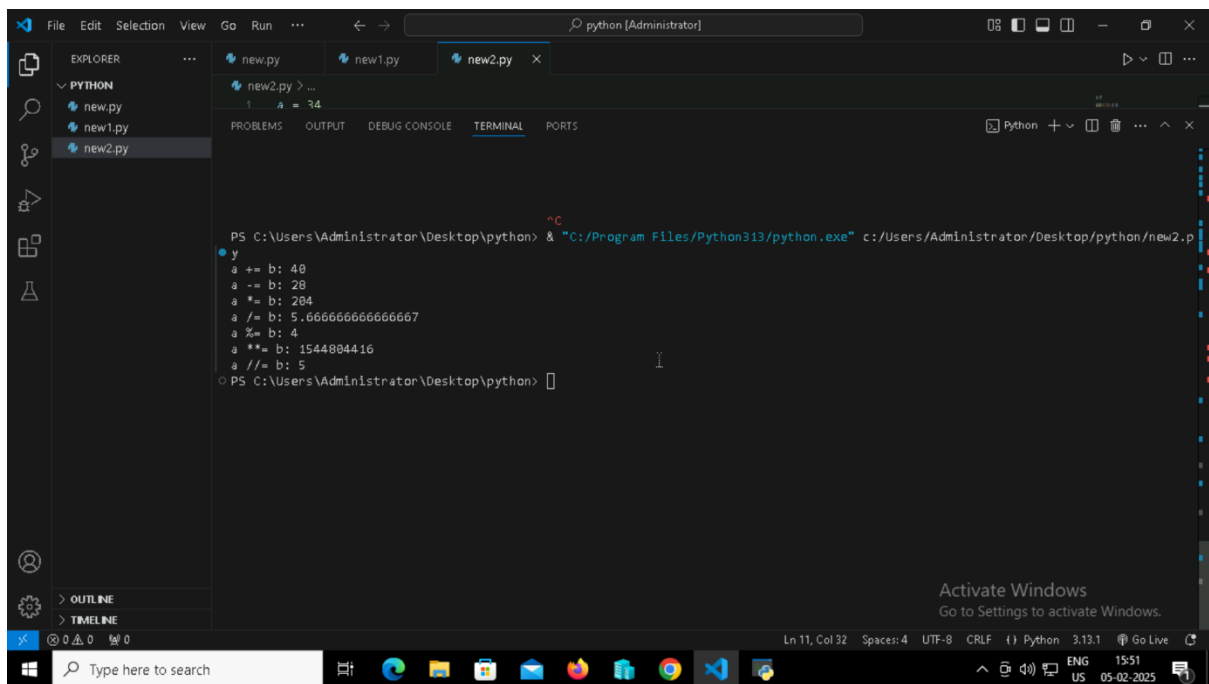
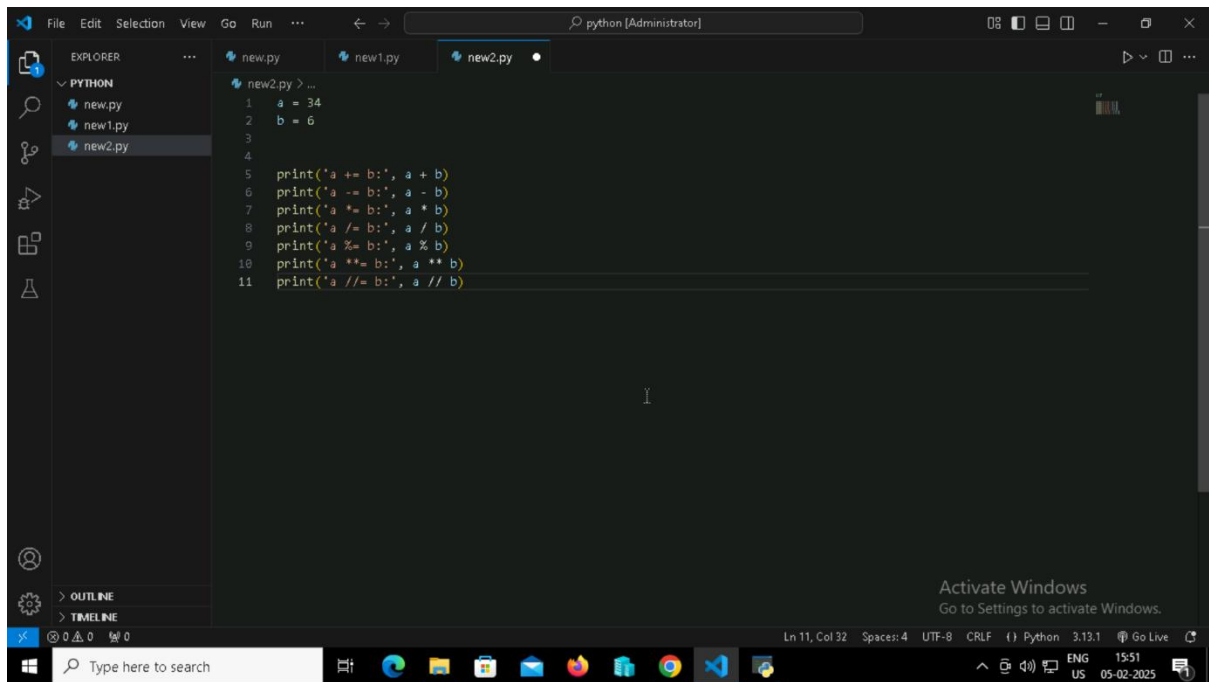
2. code examples of Comparison operators in Python:



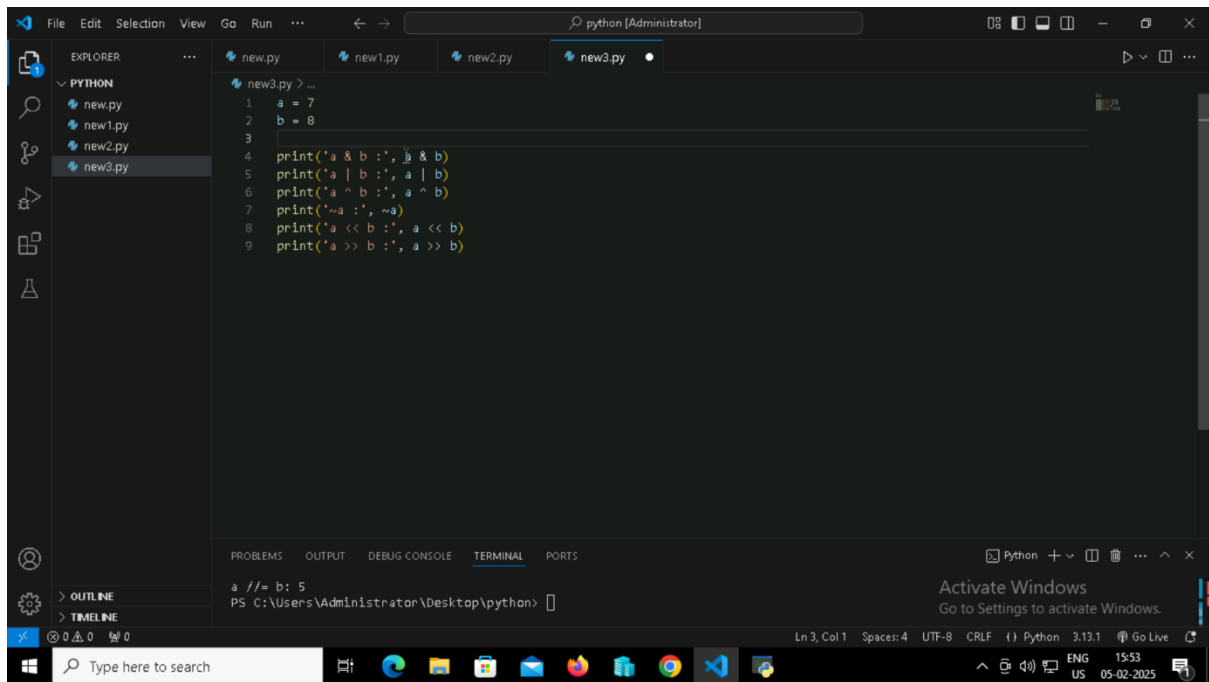
```
1 a = 46
2 b = 4
3
4 print("For a =", a, "and b =", b, "\nCheck the following:")
5
6 print('1. Two numbers are equal or not:', a == b)
7 print('2. Two numbers are not equal or not:', a != b)
8 print('3. a is less than or equal to b:', a <= b)
9 print('4. a is greater than or equal to b:', a >= b)
10 print('5. a is greater b:', a > b)
11 print('6. a is less than b:', a < b)
12
```



```
PS C:\Users\Administrator\Desktop\python> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python/new1.py
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
PS C:\Users\Administrator\Desktop\python>
```

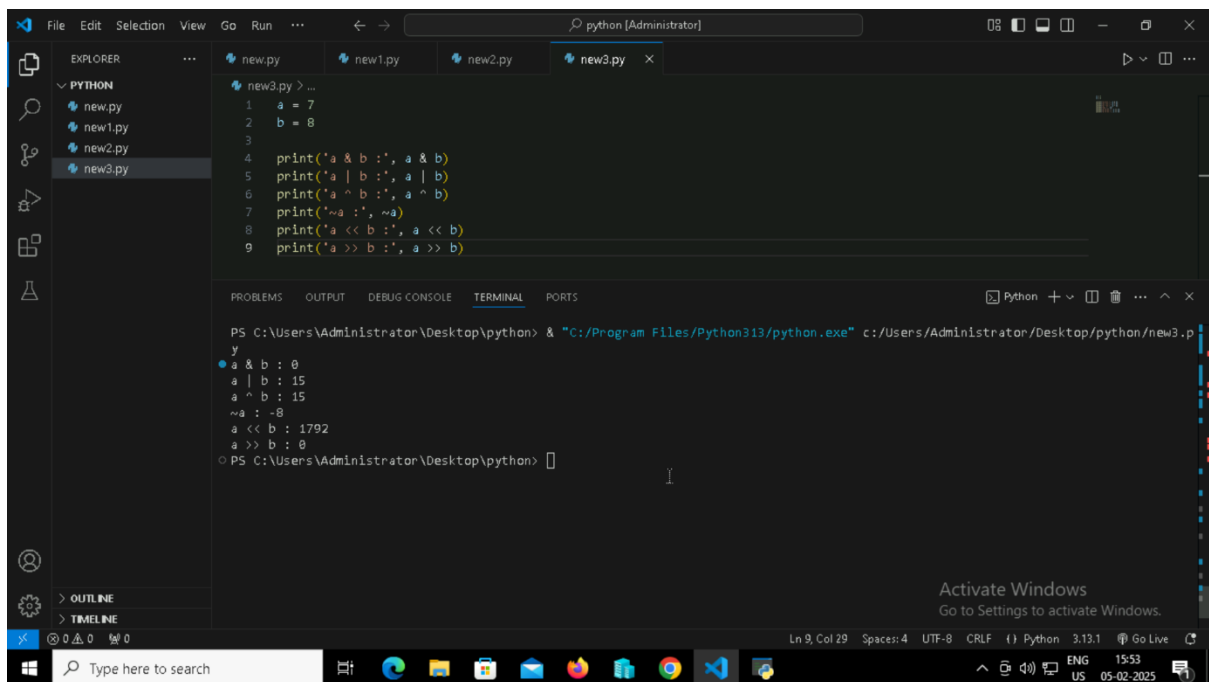


4. code examples of Logical Operators in python:



The screenshot shows the Visual Studio Code editor with a Python file named `new3.py`. The code defines two variables, `a = 7` and `b = 8`, and then prints the results of various logical operations. The Explorer sidebar on the left shows a project named `PYTHON` with files `new.py`, `new1.py`, `new2.py`, and `new3.py`. The bottom status bar indicates the file is at line 3, column 1, using UTF-8 encoding with CRLF line endings. The Windows taskbar at the bottom shows the search bar and several application icons.

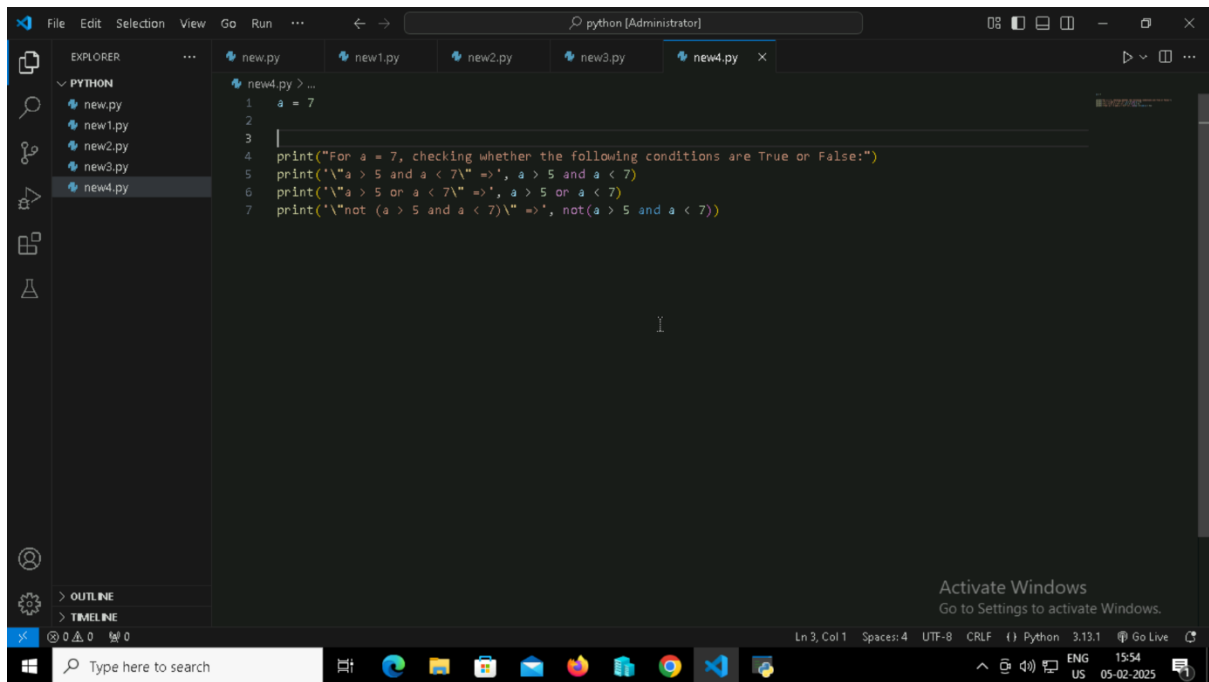
```
1 a = 7
2 b = 8
3
4 print('a & b :', a & b)
5 print('a | b :', a | b)
6 print('a ^ b :', a ^ b)
7 print('~a :', ~a)
8 print('a << b :', a << b)
9 print('a >> b :', a >> b)
```



This screenshot shows the same VS Code editor with the `new3.py` file, but the `TERMINAL` tab is active, displaying the output of the script. The command prompt shows the execution of `python new3.py` from the directory `C:\Users\Administrator\Desktop\python`. The output displays the results of the logical operations performed on `a = 7` and `b = 8`. The status bar now shows the cursor at line 9, column 29. The Windows taskbar remains visible at the bottom.

```
PS C:\Users\Administrator\Desktop\python> "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python/new3.py
y
a & b : 0
a | b : 15
a ^ b : 15
~a : -8
a << b : 1792
a >> b : 8
PS C:\Users\Administrator\Desktop\python>
```

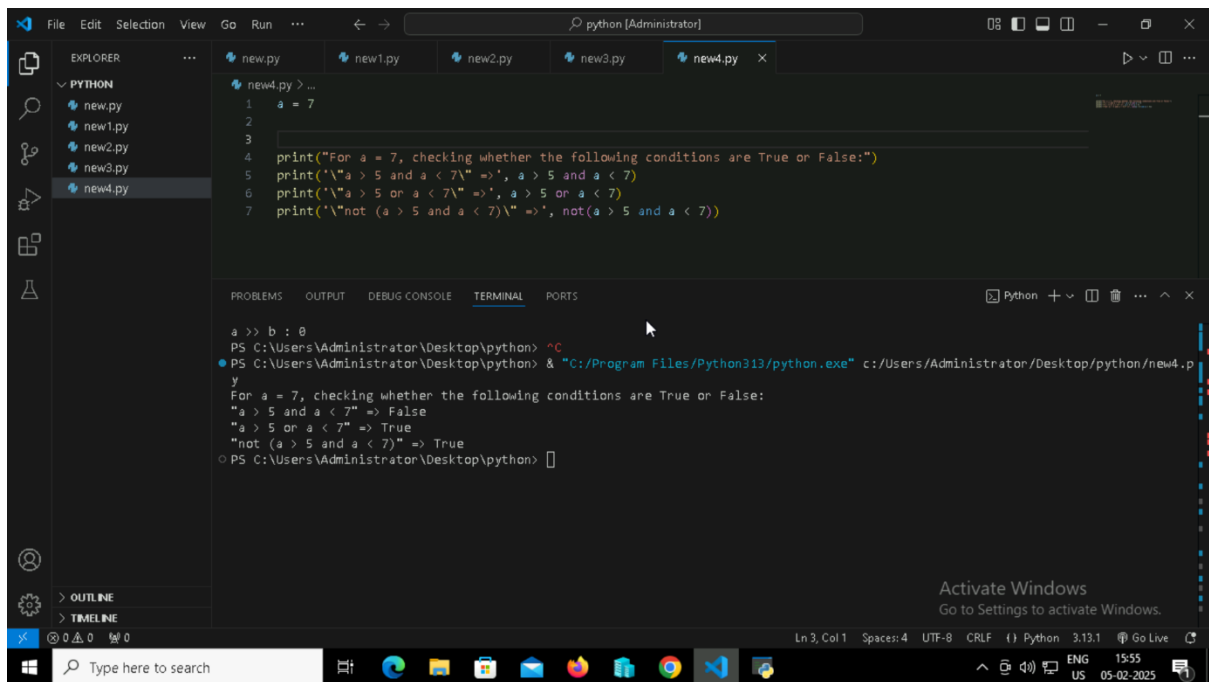
5. code examples of Bitwise Operators in python:



The screenshot shows the Visual Studio Code editor with a file named `new4.py` open. The code in the editor is as follows:

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

The Explorer sidebar on the left shows a list of Python files: `new.py`, `new1.py`, `new2.py`, `new3.py`, and `new4.py`. The bottom status bar indicates the current line and column as `Ln 3, Col 1`.

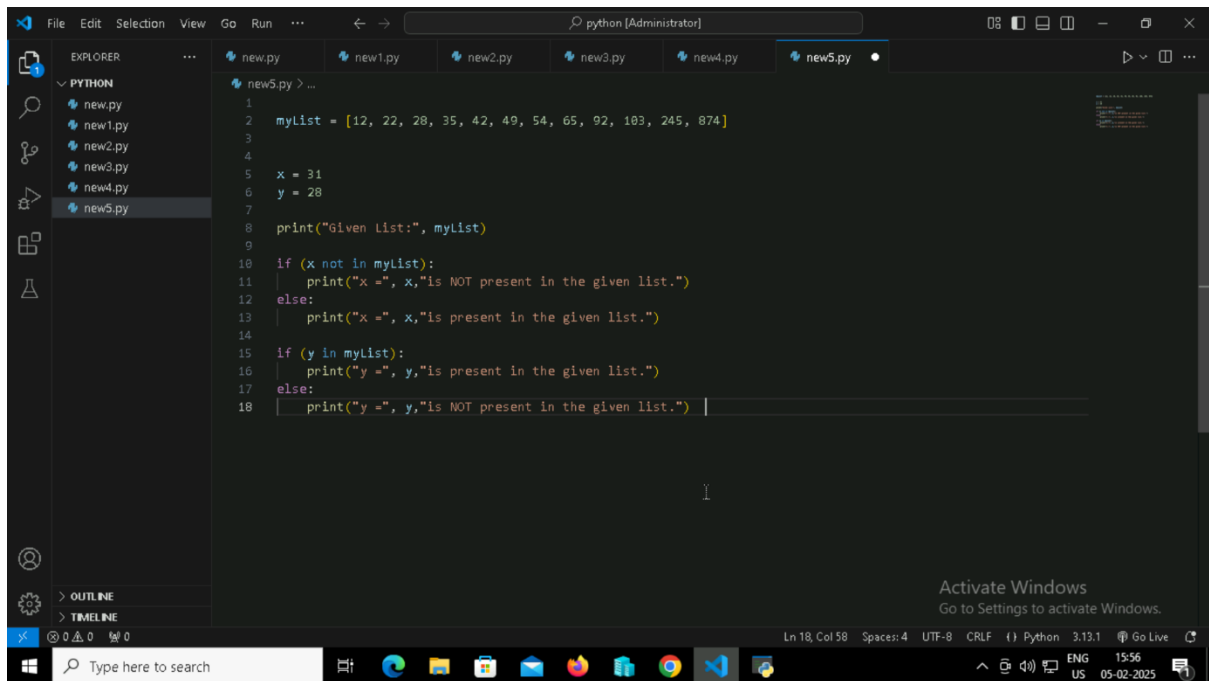


This screenshot shows the same VS Code editor with the `new4.py` file, but now the `TERMINAL` panel is active at the bottom. It displays the command to run the script and its output:

```
a >> b : 8
PS C:\Users\Administrator\Desktop\python> ^C
PS C:\Users\Administrator\Desktop\python> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python/new4.py
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
PS C:\Users\Administrator\Desktop\python> 
```

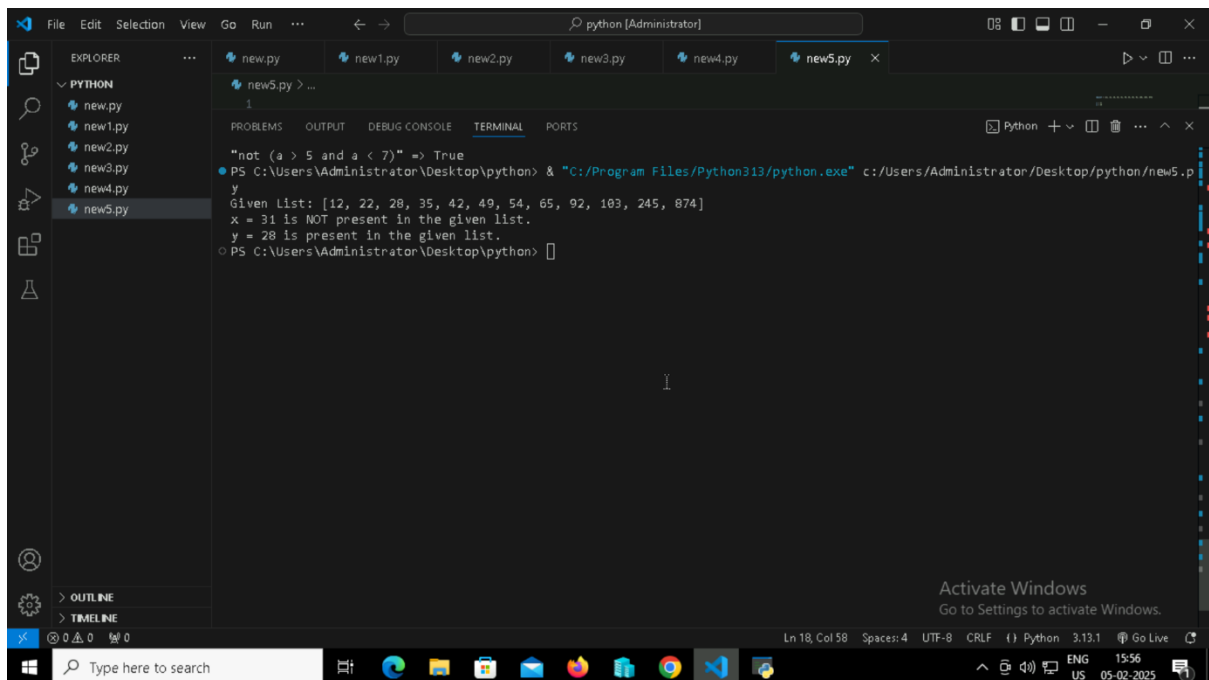
The terminal output matches the print statements in the code above. The status bar at the bottom shows `Ln 3, Col 1` and `Spaces: 4`.

6. code examples of Membership Operators in python:



The screenshot shows the Visual Studio Code editor with a Python file named `new5.py`. The code defines a list `myList` and checks for the presence of variables `x` and `y` using membership operators.

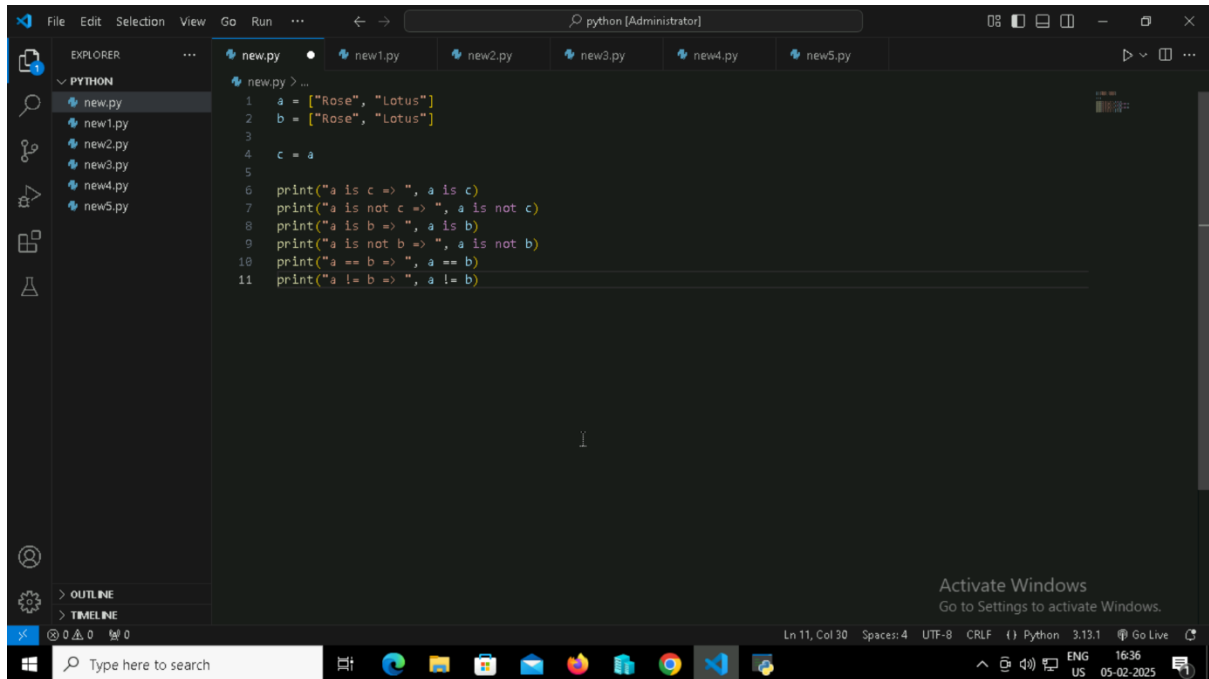
```
1 myList = [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]
2
3
4
5 x = 31
6 y = 28
7
8 print("Given list:", myList)
9
10 if (x not in myList):
11     print("x =", x, "is NOT present in the given list.")
12 else:
13     print("x =", x, "is present in the given list.")
14
15 if (y in myList):
16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



The screenshot shows the same VS Code editor with the `new5.py` file. The `TERMINAL` tab is active, displaying the output of the script execution.

```
PS C:\Users\Administrator\Desktop\python> "C:\Program Files\Python313\python.exe" c:/Users/Administrator/Desktop/python/new5.py
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.
```

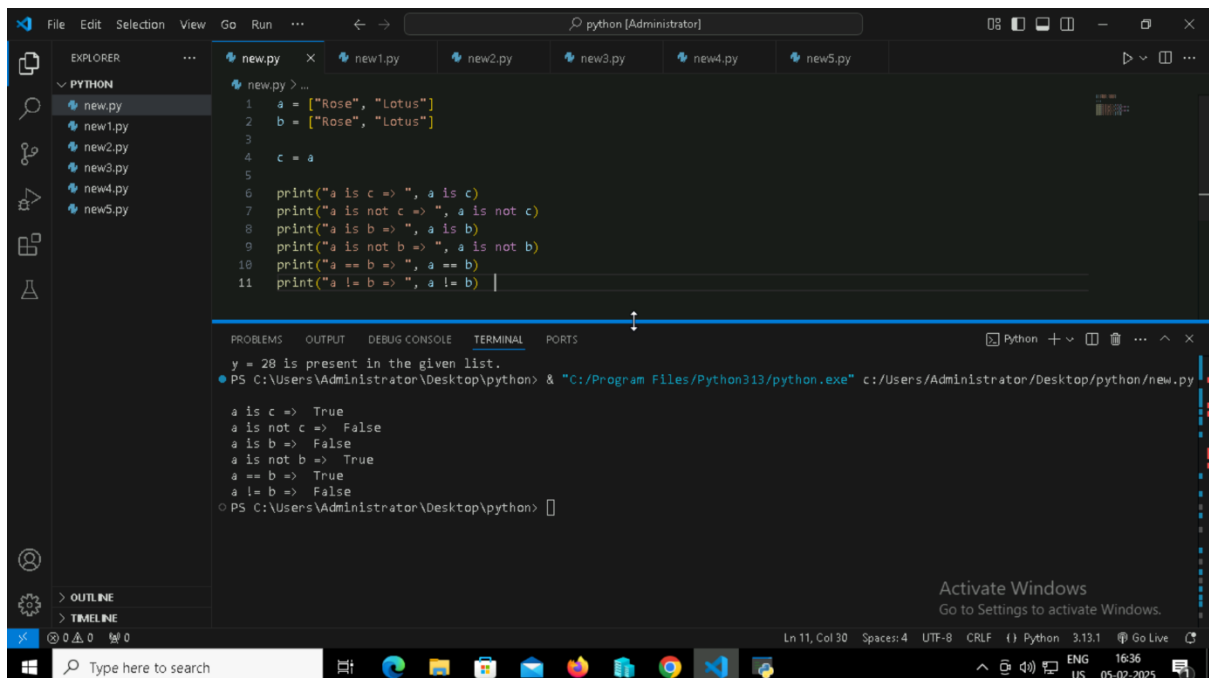
7.. code examples of Identity Operators in python:



The screenshot shows the Visual Studio Code editor with a Python file named `new.py`. The code defines two lists, `a` and `b`, both containing the elements "Rose" and "Lotus". It then assigns `c = a`. The script uses identity operators (`is`, `is not`) and equality operators (`==`, `!=`) to compare the variables. The output of the script is as follows:

```
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)
```

The status bar at the bottom indicates the file is at line 11, column 30, using UTF-8 encoding with CRLF line endings. The Python version is 3.13.1.

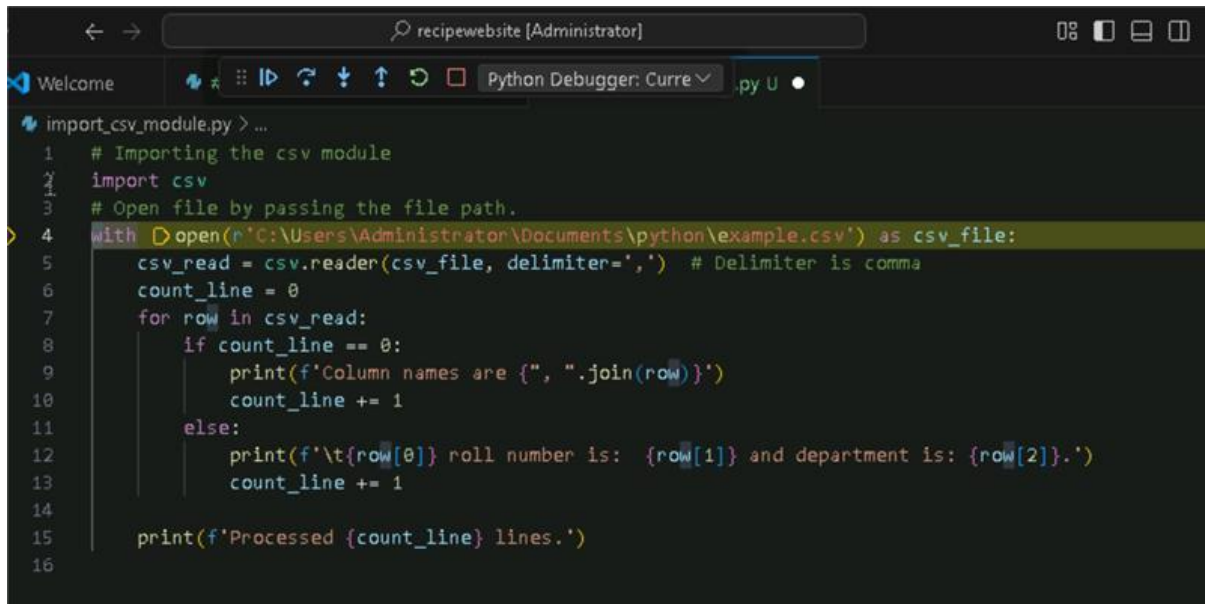


The screenshot shows the same Visual Studio Code editor with the `new.py` file. The `TERMINAL` panel is open, displaying the output of the script. The output confirms that `a` and `c` are the same object in memory, while `a` and `b` are different objects. The output is as follows:

```
y = 28 is present in the given list.
PS C:\Users\Administrator\Desktop\python> "C:\Program Files\Python313\python.exe" c:/Users/Administrator/Desktop/python/new.py
a is c => True
a is not c => False
a is b => False
a is not b => True
a == b => True
a != b => False
PS C:\Users\Administrator\Desktop\python>
```

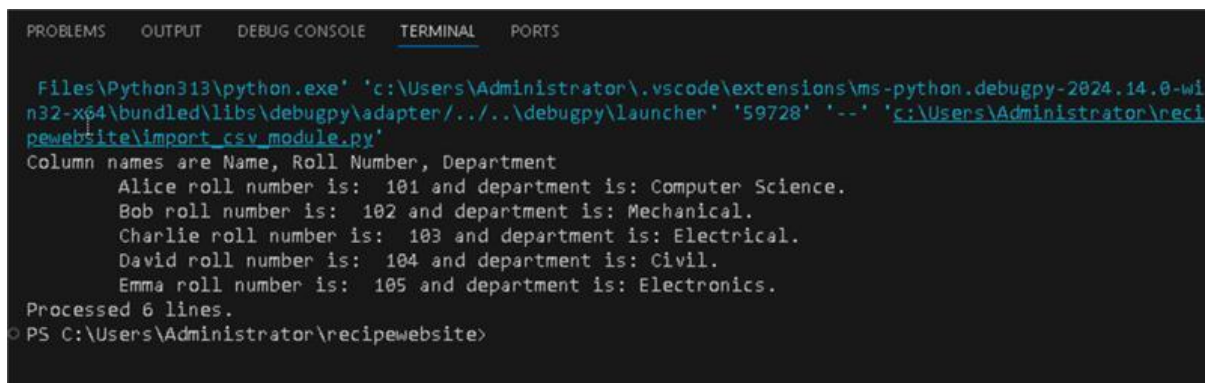
The status bar at the bottom indicates the file is at line 11, column 30, using UTF-8 encoding with CRLF line endings. The Python version is 3.13.1.

8. To Read CSV file in Python



The screenshot shows the Visual Studio Code editor with a Python file named `import_csv_module.py`. The code is as follows:

```
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
```



The screenshot shows the terminal output of the Python script. The command executed is:

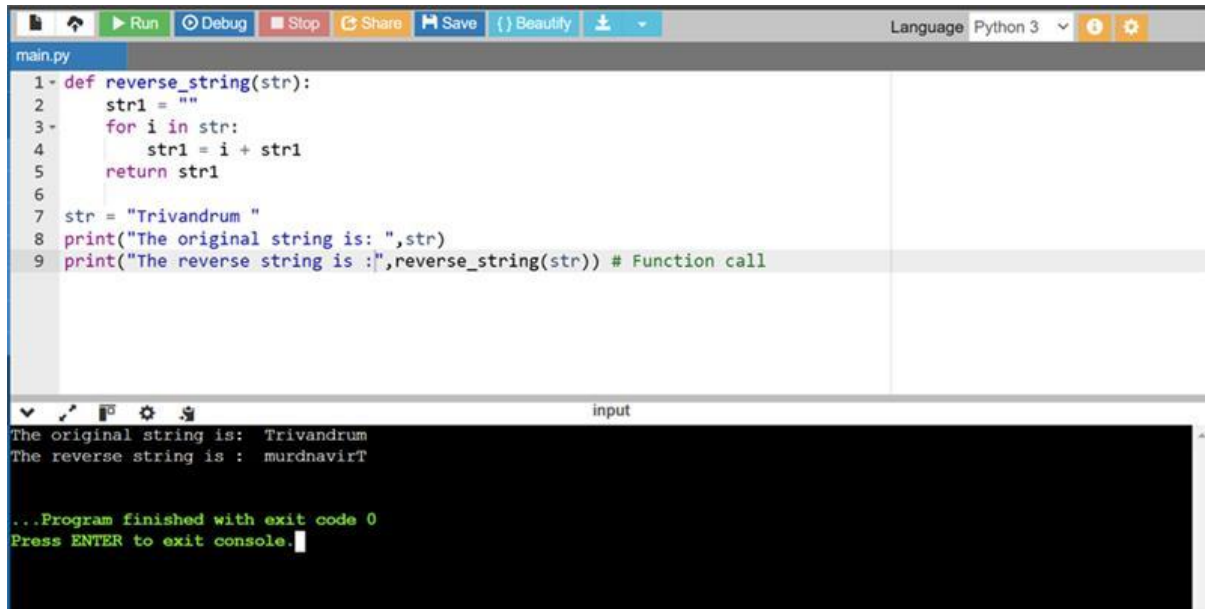
```
Files\Python313\python.exe 'c:\Users\Administrator\.vscode\extensions\ms-python.debugpy-2024.14.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher' '59728' '--' 'c:\Users\Administrator\recipewebsite\import_csv_module.py'
```

The output is as follows:

```
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 iterates over each character in a string and builds a reversed string. It then prints the original and reversed strings. The output in the console shows the original string 'Trivandrum' and the reversed string 'murdnaviT'.

```
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 : murdnaviT

...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 uses a while loop to reverse a string by iterating from the end to the beginning. It then prints the original and reversed strings. The output in the console shows the original string 'Trivandrum' and the reversed string 'murdnaviT'.

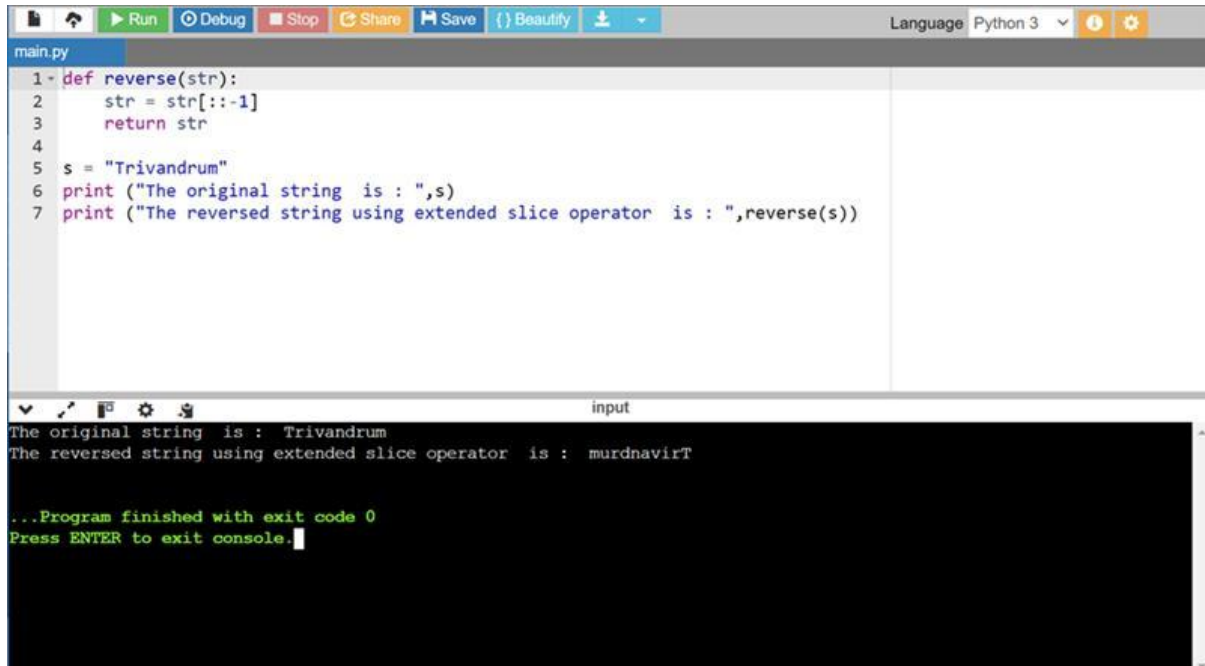
```
1 # Reverse string
2 # Using a while loop
3
4 str = "Trivandrum"
5 print ("The original string is : ",str)
6 reverse_String = ""
7 count = len(str)
8- while count > 0:
9     reverse_String += str[ count - 1 ]
10     count = count - 1
11 print ("The reversed string using a while loop is : ",reverse_String)#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

input

The original string is : Trivandrum
The reversed string using a while loop is : murdnaviT

...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`. The code defines a function `reverse(str)` that returns the reversed string using the slice operator `str[::-1]`. It then uses this function to reverse the string "Trivandrum".

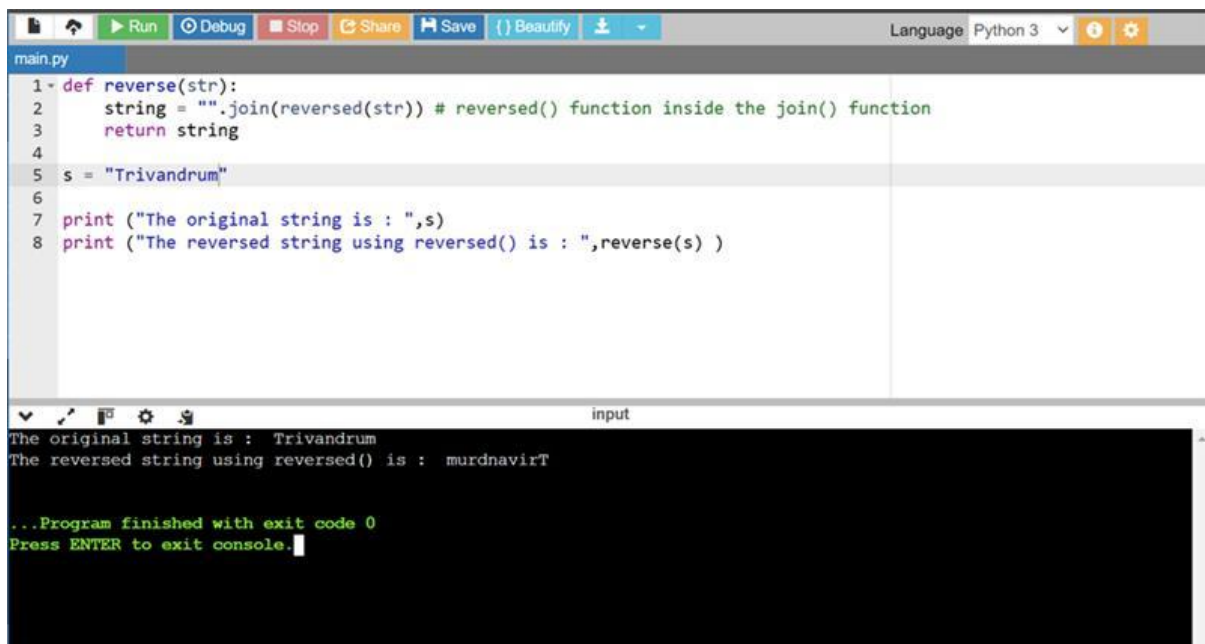
```
1- def reverse(str):
2-     str = str[::-1]
3-     return str
4-
5- s = "Trivandrum"
6- print ("The original string is : ",s)
7- print ("The reversed string using extended slice operator is : ",reverse(s))
```

The output in the console is:

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

...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`. The code defines a function `reverse(str)` that returns the reversed string using the `reversed()` function inside a `join()` function. It then uses this function to reverse the string "Trivandrum".

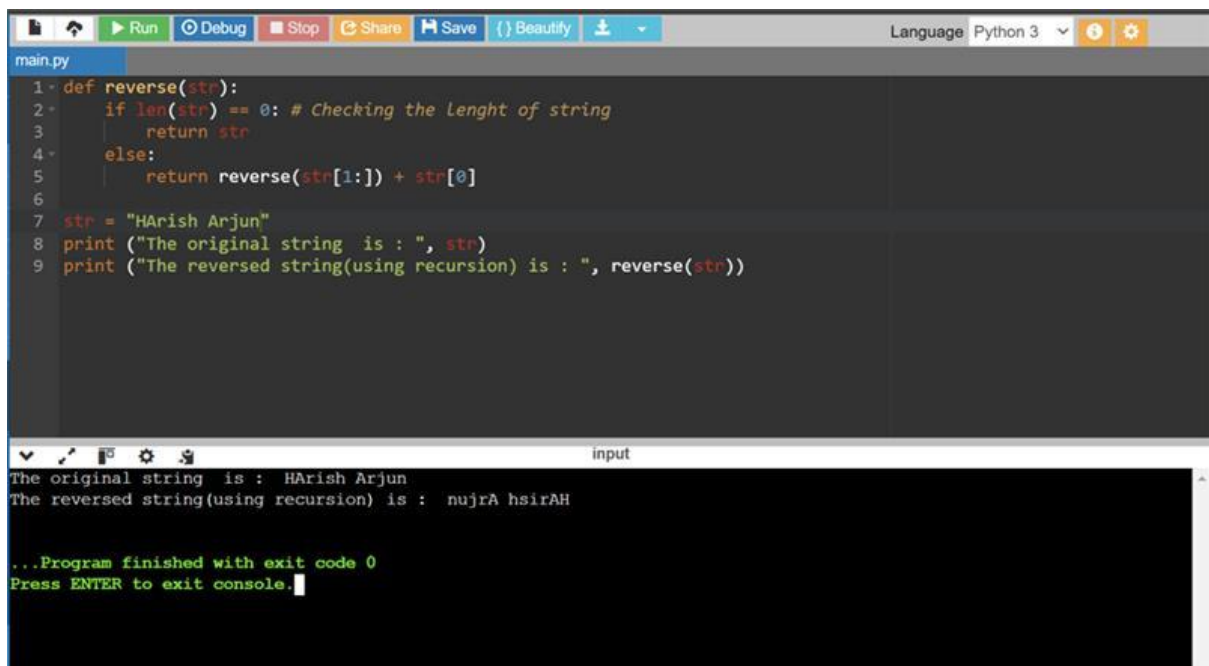
```
1- def reverse(str):
2-     string = "".join(reversed(str)) # reversed() function inside the join() function
3-     return string
4-
5- s = "Trivandrum"
6-
7- print ("The original string is : ",s)
8- print ("The reversed string using reversed() is : ",reverse(s) )
```

The output in the console is:

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

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

5. Using the Recursion



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

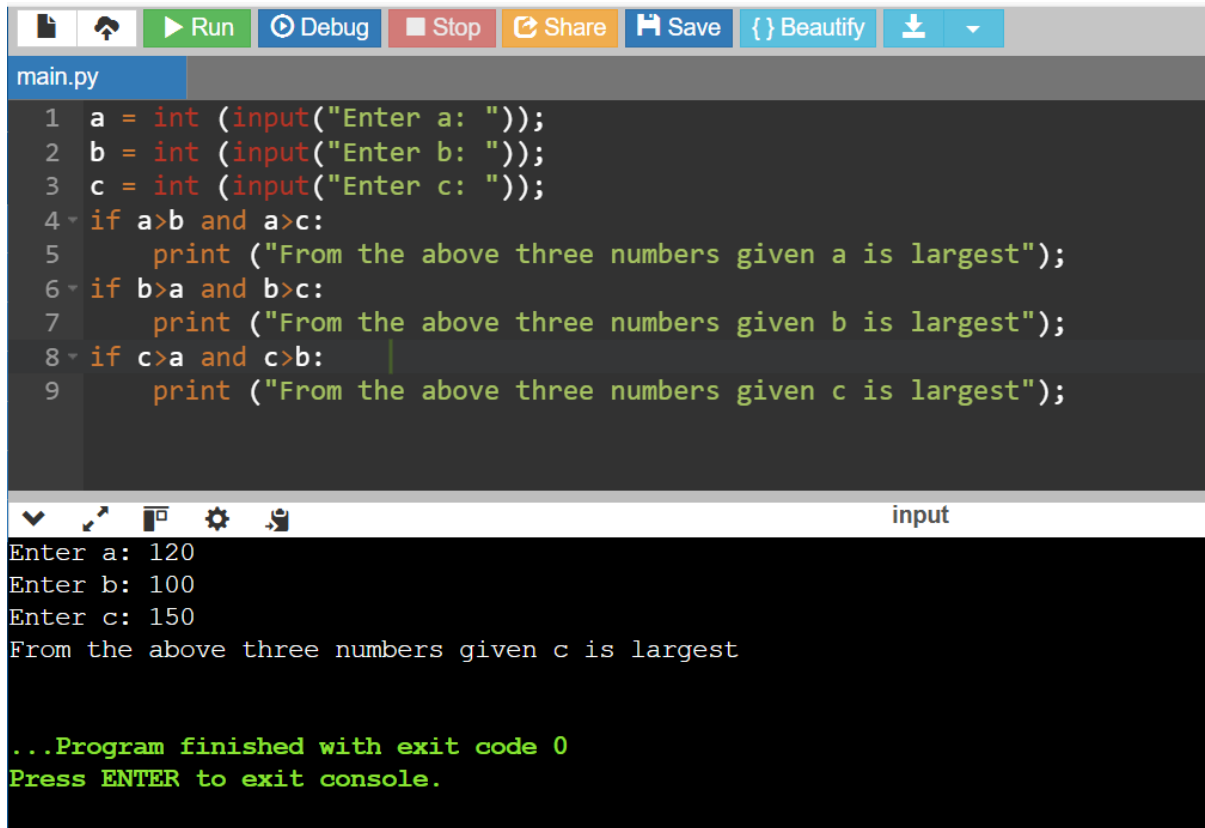
```
1- def reverse(str):  
2-     if len(str) == 0: # Checking the Lenght of string  
3-         return str  
4-     else:  
5-         return reverse(str[1:]) + str[0]  
6-  
7- str = "HArish Arjun"  
8- print ("The original string is : ", str)  
9- print ("The reversed string(using recursion) is :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

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

```
The original string is : HArish Arjun  
The reversed string(using recursion) is : nujrA hsirAH  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

If Statement:

Example 1:



The screenshot shows a Python IDE with a toolbar at the top containing icons for file operations, a 'Run' button, 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. The editor window, titled 'main.py', contains the following code:

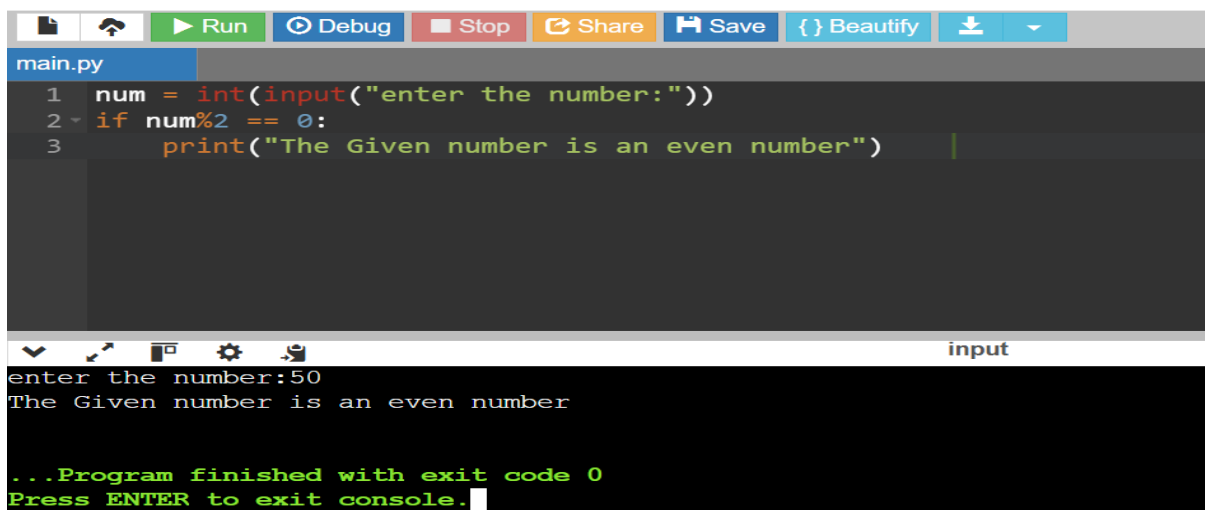
```
1 a = int (input("Enter a: "));
2 b = int (input("Enter b: "));
3 c = int (input("Enter c: "));
4 if a>b and a>c:
5     print ("From the above three numbers given a is largest");
6 if b>a and b>c:
7     print ("From the above three numbers given b is largest");
8 if c>a and c>b:
9     print ("From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

Below the editor is a console window titled 'input'. It displays the program's execution with the following text:

```
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:



The screenshot shows a Python IDE with a toolbar at the top containing icons for file operations, a 'Run' button, 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. The editor window, titled 'main.py', contains the following code:

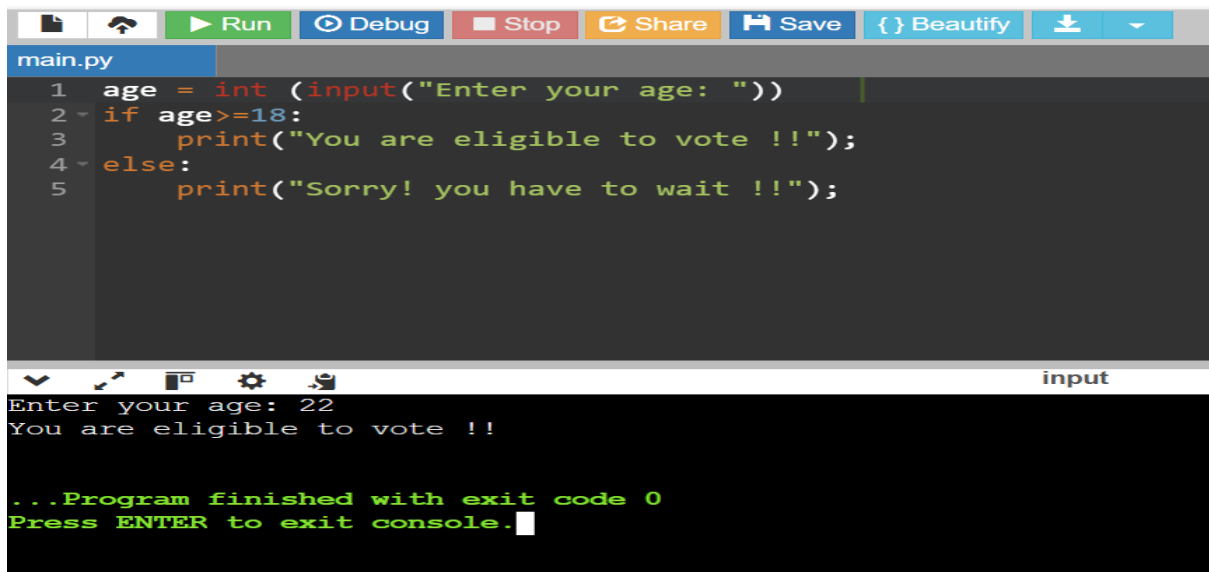
```
1 num = int(input("enter the number:"))
2 if num%2 == 0:
3     print("The Given number is an even number")
```

Below the editor is a console window titled 'input'. It displays the program's execution with the following text:

```
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:

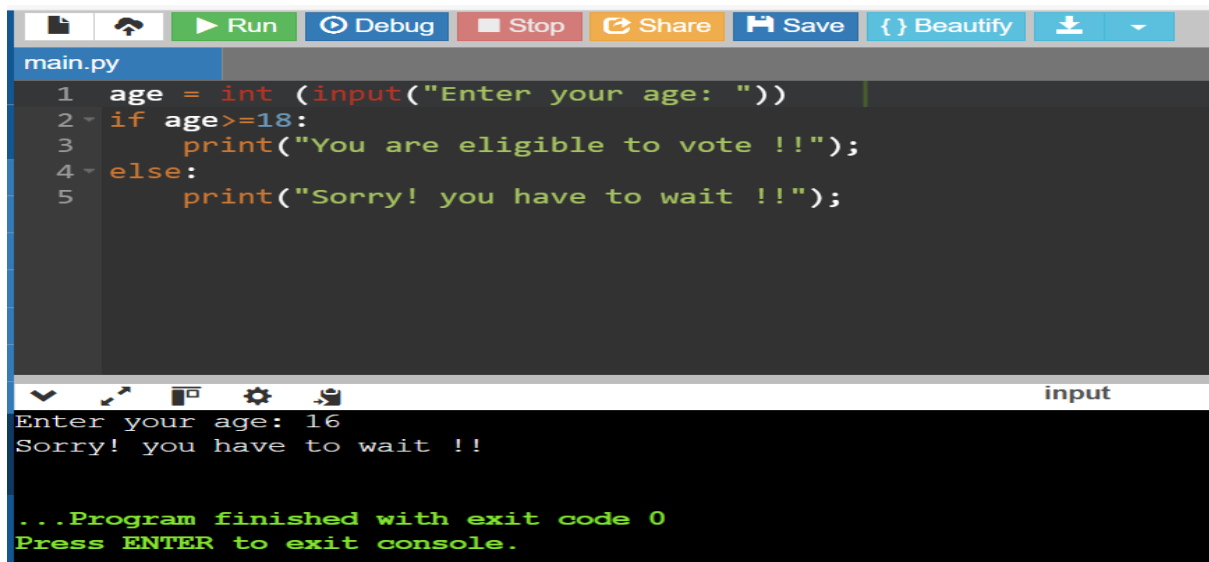


```
main.py
1 age = int (input("Enter your age: "))
2 if age>=18:
3     print("You are eligible to vote !!");
4 else:
5     print("Sorry! you have to wait !!");
```

input

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

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



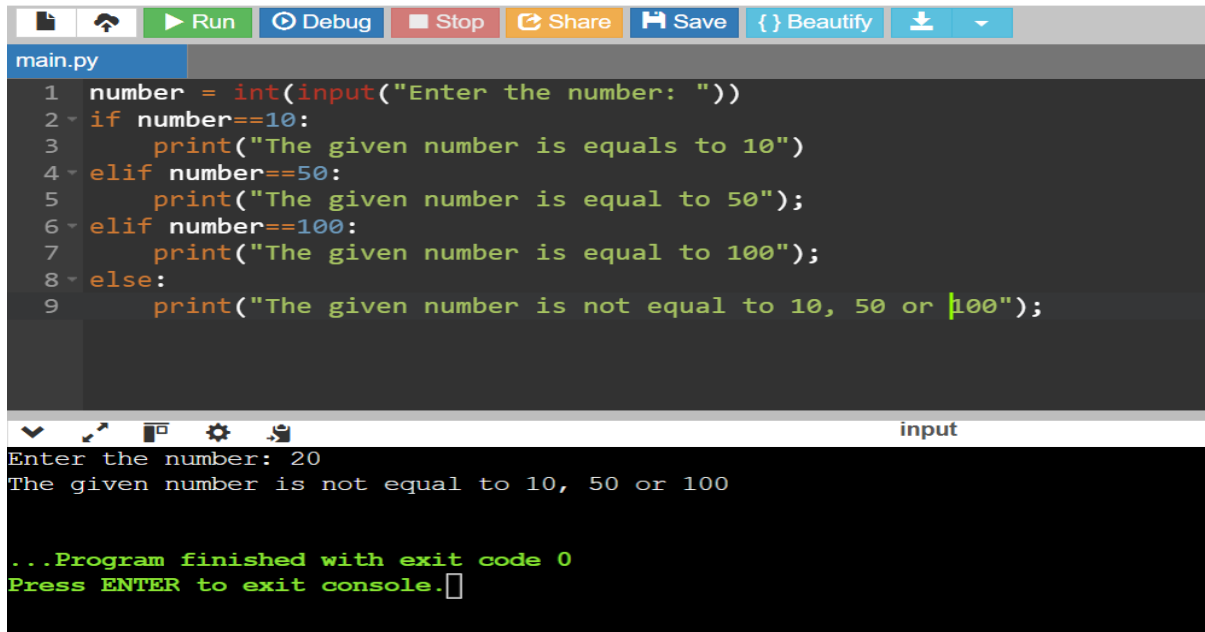
```
main.py
1 age = int (input("Enter your age: "))
2 if age>=18:
3     print("You are eligible to vote !!");
4 else:
5     print("Sorry! you have to wait !!");
```

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 code editor with a toolbar at the top containing icons for file operations, a 'Run' button, 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. The file name 'main.py' is displayed. The code in the editor is as follows:

```
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");
```

Below the code editor, the console output is shown. It indicates that the user entered '20' and the program printed 'The given number is not equal to 10, 50 or 100'. The program finished with exit code 0.

```
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



The screenshot shows a code editor with the same toolbar as the first image. The file name 'main.py' is displayed. The code in the editor is as follows:

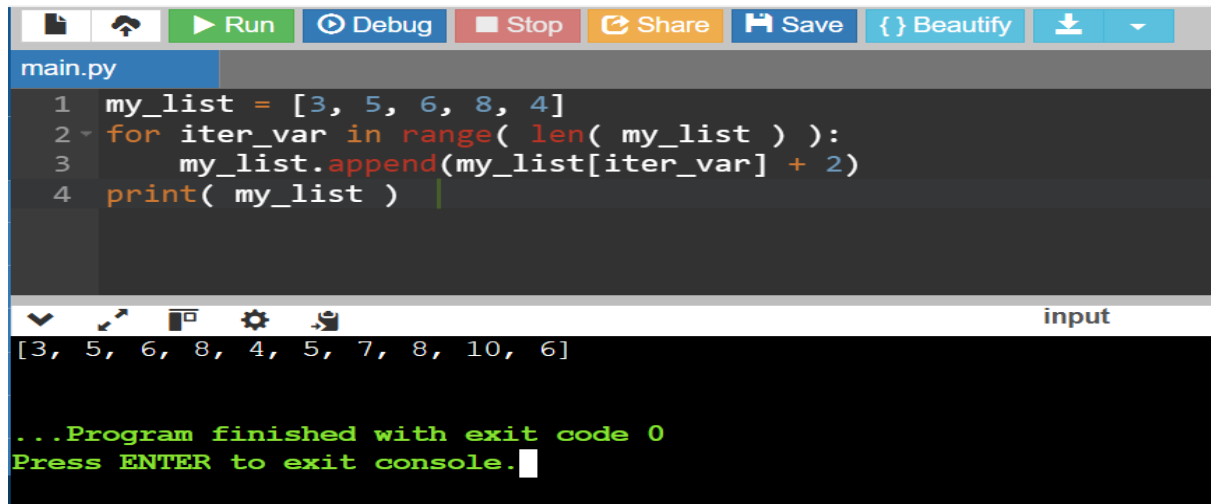
```
1 numbers = [3, 5, 23, 6, 5, 1, 2, 9, 8]
2 sum_ = 0
3 for num in numbers:
4     sum_ = sum_ + num ** 2
5 print("The sum of squares is: ", sum_)
```

Below the code editor, the console output is shown. It indicates that the program printed 'The sum of squares is: 774'. The program finished with exit code 0.

```
The sum of squares is: 774

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

2. Using Range ()



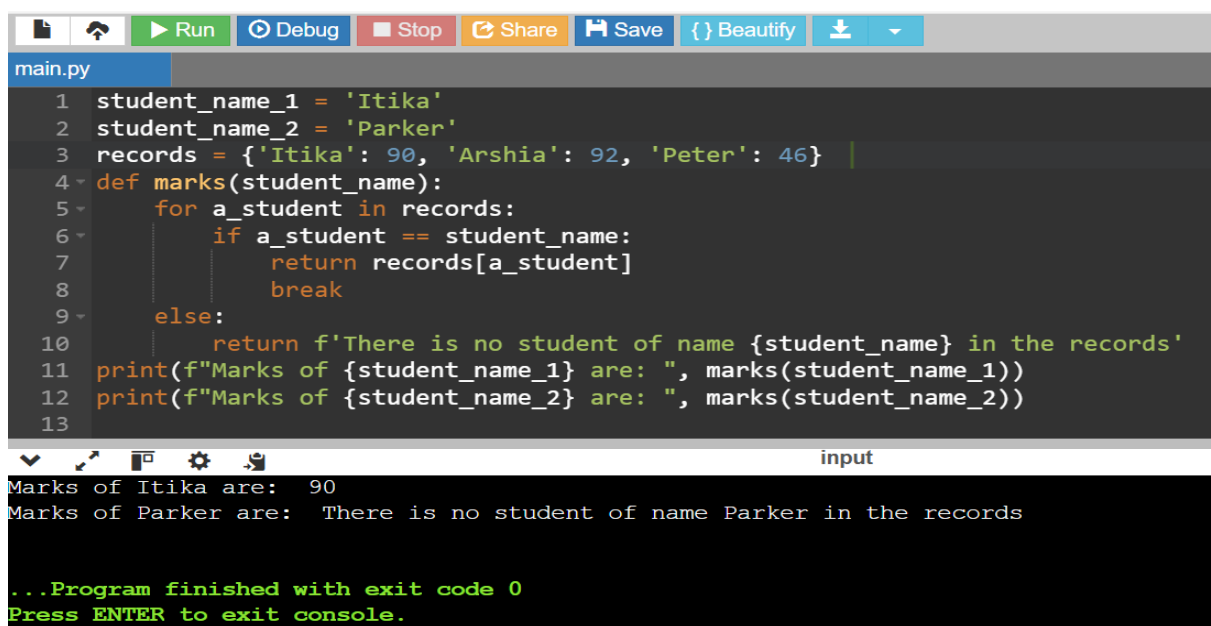
```
main.py
1 my_list = [3, 5, 6, 8, 4]
2 for iter_var in range( len( my_list ) ):
3     my_list.append(my_list[iter_var] + 2)
4 print( my_list )
```

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 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

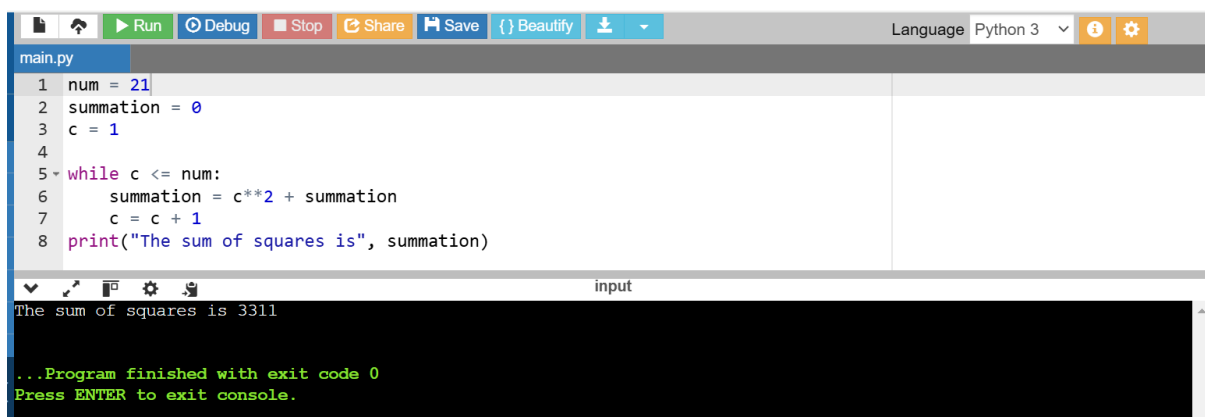


```
main.py
1 import random
2 numbers = [ ]
3 for val in range(0, 11):
4     numbers.append( random.randint( 0, 11 ) )
5 for num in range( 0, 11 ):
6     for i in numbers:
7         if num == i:
8             print( num, end = " " )

0 1 2 3 4 5 6 7 9 10
...Program finished with exit code 0
Press ENTER to exit console.
```

WHILE Loops:

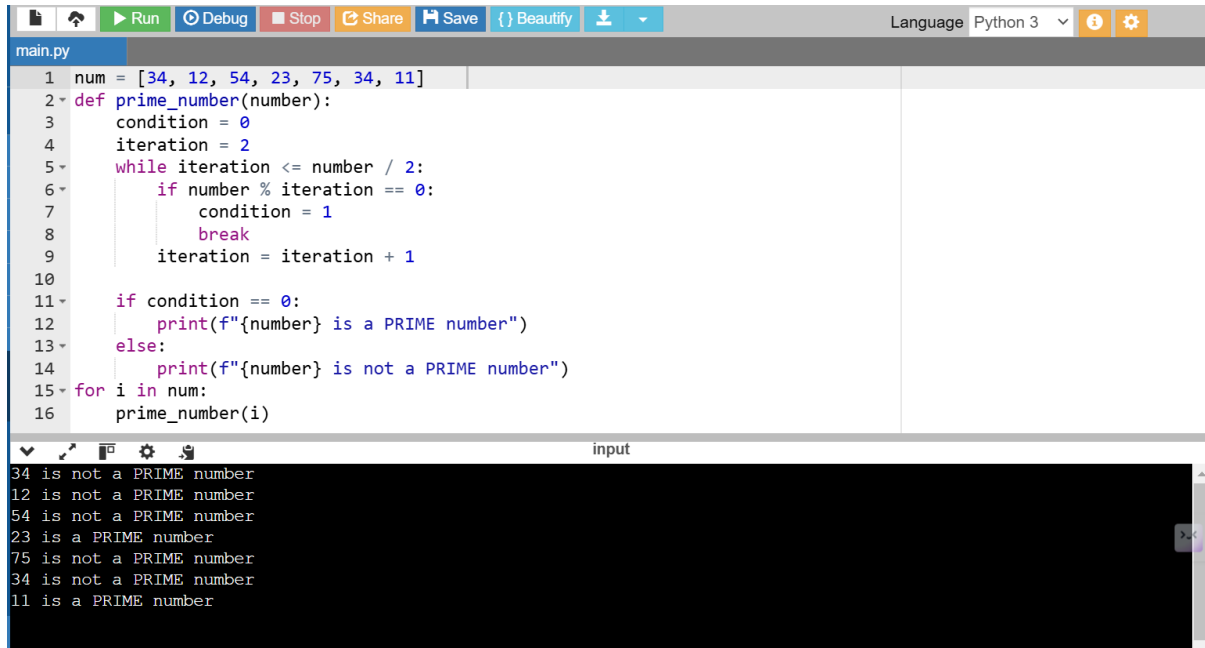
1. Sum of squares



```
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)

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



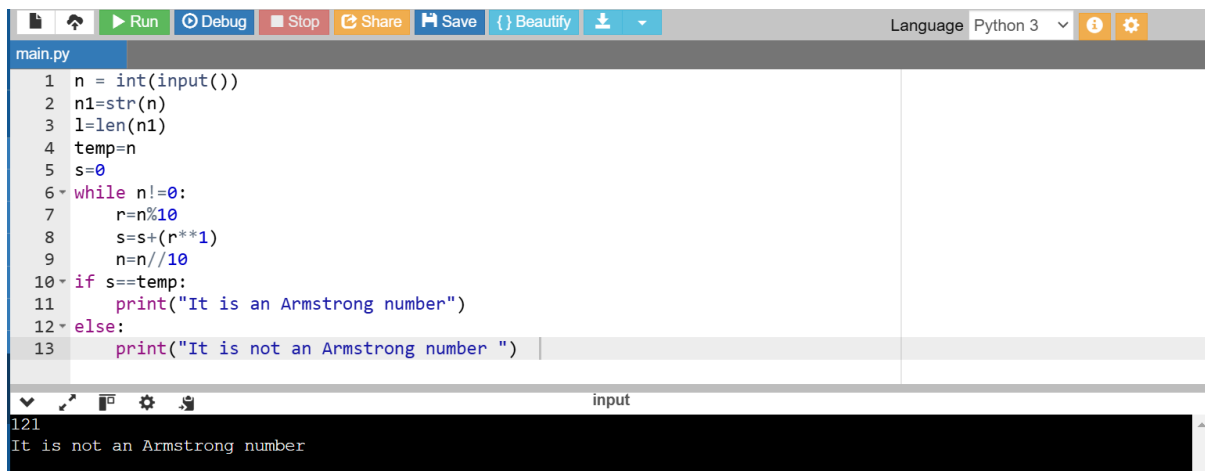
The screenshot shows a Python IDE with a file named `main.py`. The code defines a function `prime_number(number)` that checks if a number is prime. It uses a `while` loop to test divisibility from 2 up to the number. If the number is divisible by any of these, it's not prime. Otherwise, it is prime. The program then iterates over a list of numbers: `[34, 12, 54, 23, 75, 34, 11]`.

```
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)
```

The output window shows the results of the program's execution:

```
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



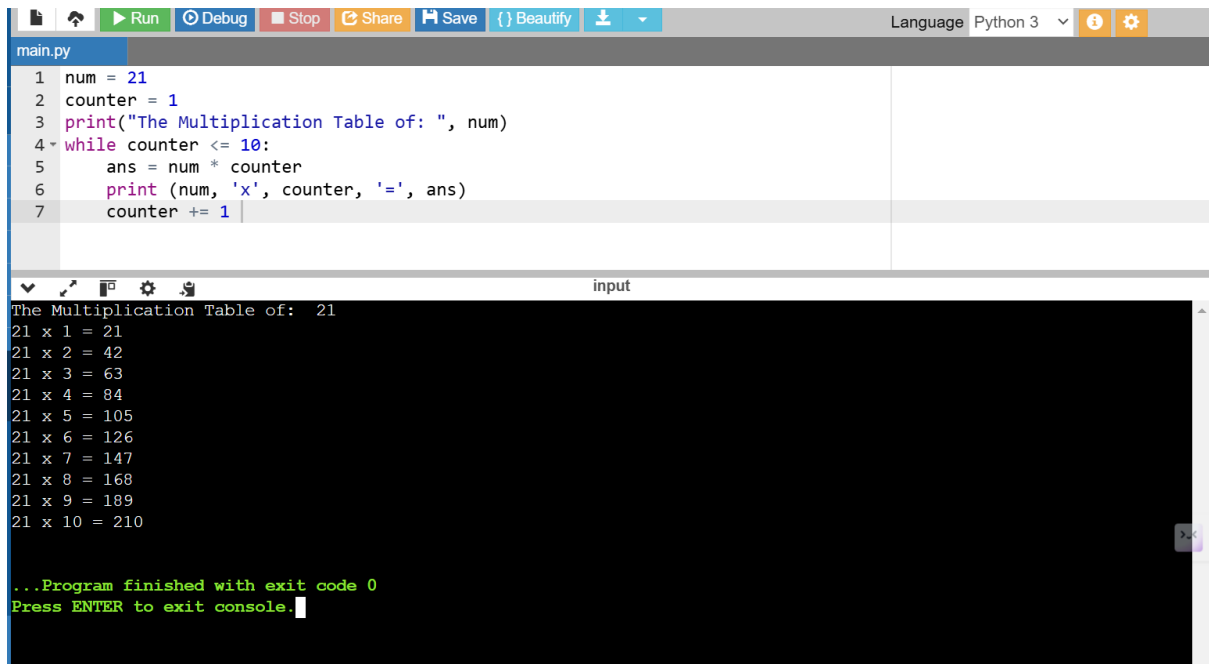
The screenshot shows a Python IDE with a file named `main.py`. The code takes an input number `n`, converts it to a string `n1`, and calculates its length `l`. It then uses a `while` loop to calculate the sum of the cubes of its digits. If the sum equals the original number, it's an Armstrong number.

```
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 ")
```

The output window shows the result of the program's execution:

```
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:

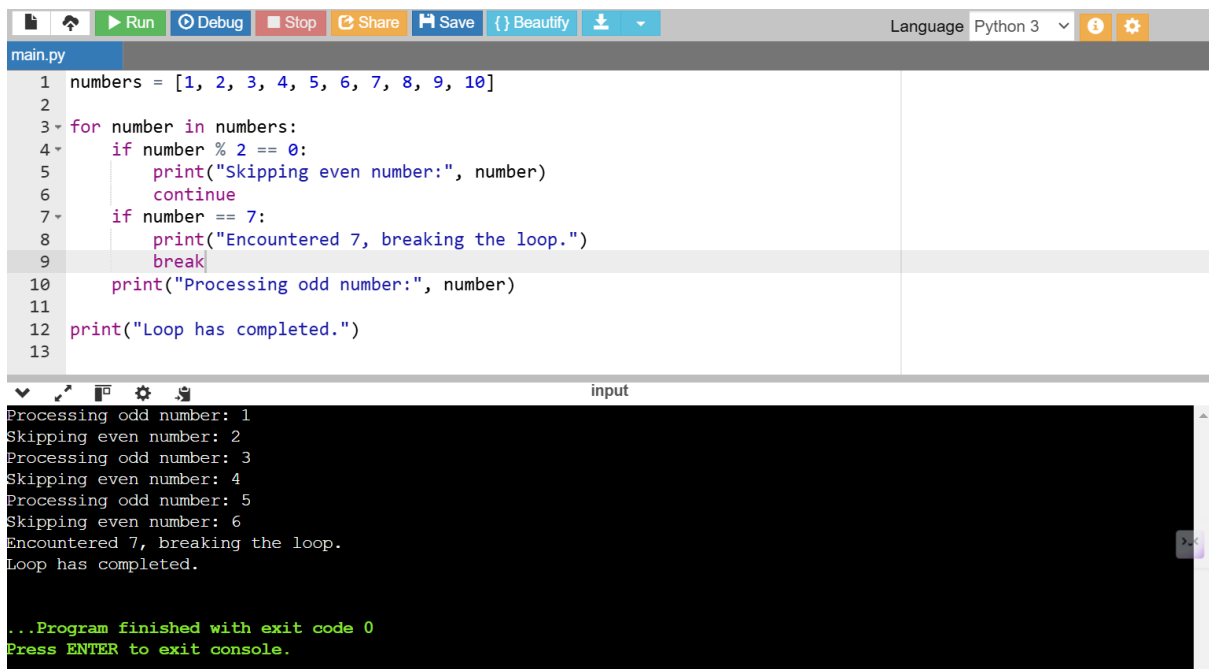
```
1 num = 21
2 counter = 1
3 print("The Multiplication Table of: ", num)
4 while counter <= 10:
5     ans = num * counter
6     print (num, 'x', counter, '=', ans)
7     counter += 1
```

The output window shows the following text:

```
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 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 window shows the following text:

```
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.
```

