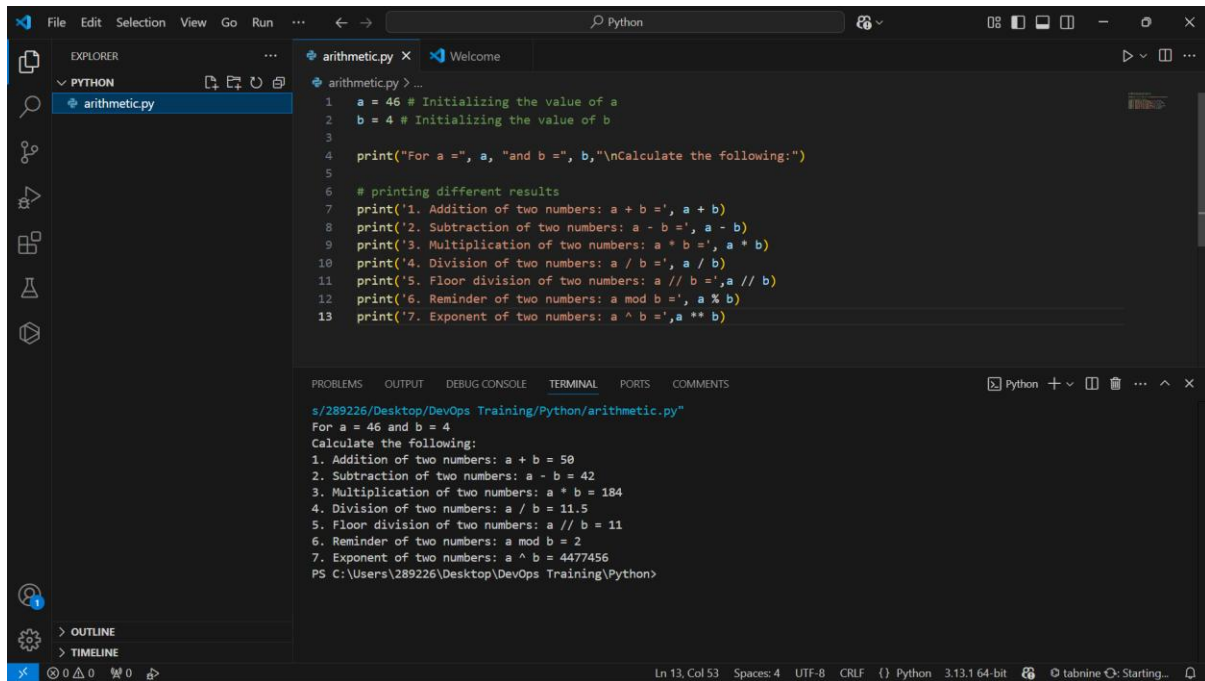


PYTHON ASSIGNMENT

ARITHMETIC OPERATORS:

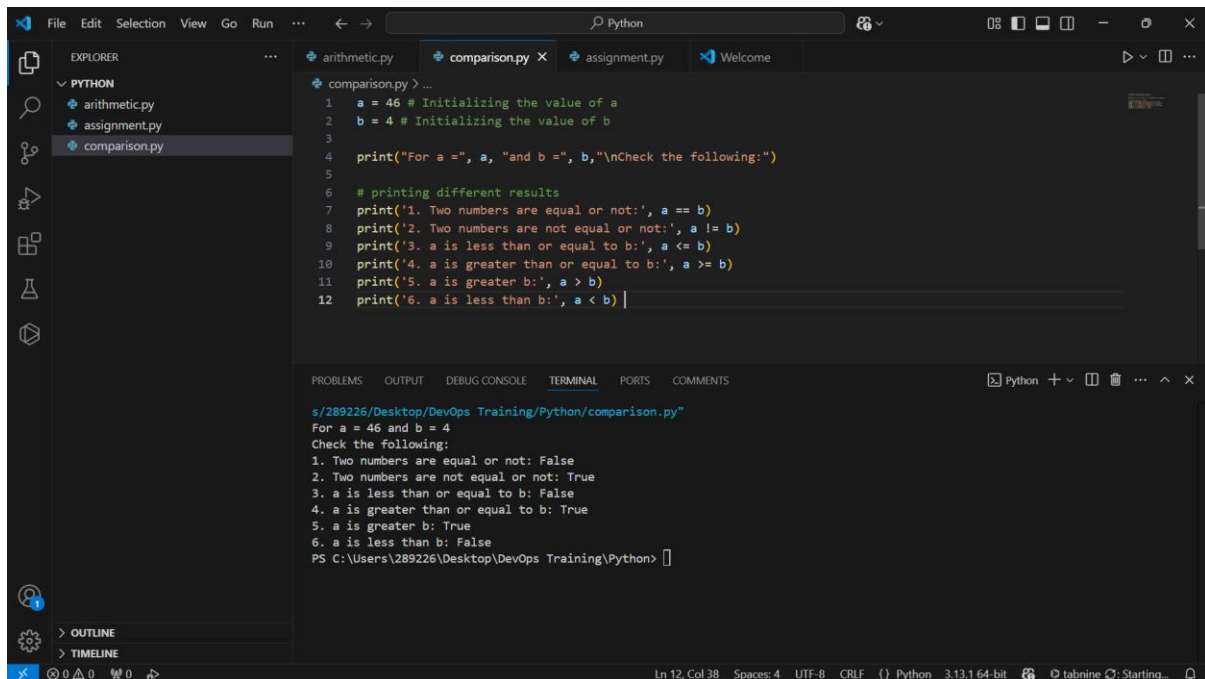


The screenshot shows a Visual Studio Code editor with a file named `arithmetic.py` open. The code defines two variables, `a = 46` and `b = 4`, and prints various arithmetic results. The terminal output shows the execution of the script, displaying the calculated values for each operation.

```
1 a = 46 # Initializing the value of a
2 b = 4 # Initializing the value of b
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. Remainder of two numbers: a mod b =', a % b)
13 print('7. Exponent of two numbers: a ^ b =', a ** b)
```

```
s/289226/Desktop/DevOps Training/Python/arithmetic.py
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. Remainder of two numbers: a mod b = 2
7. Exponent of two numbers: a ^ b = 4477456
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

COMPARISON OPERATORS:

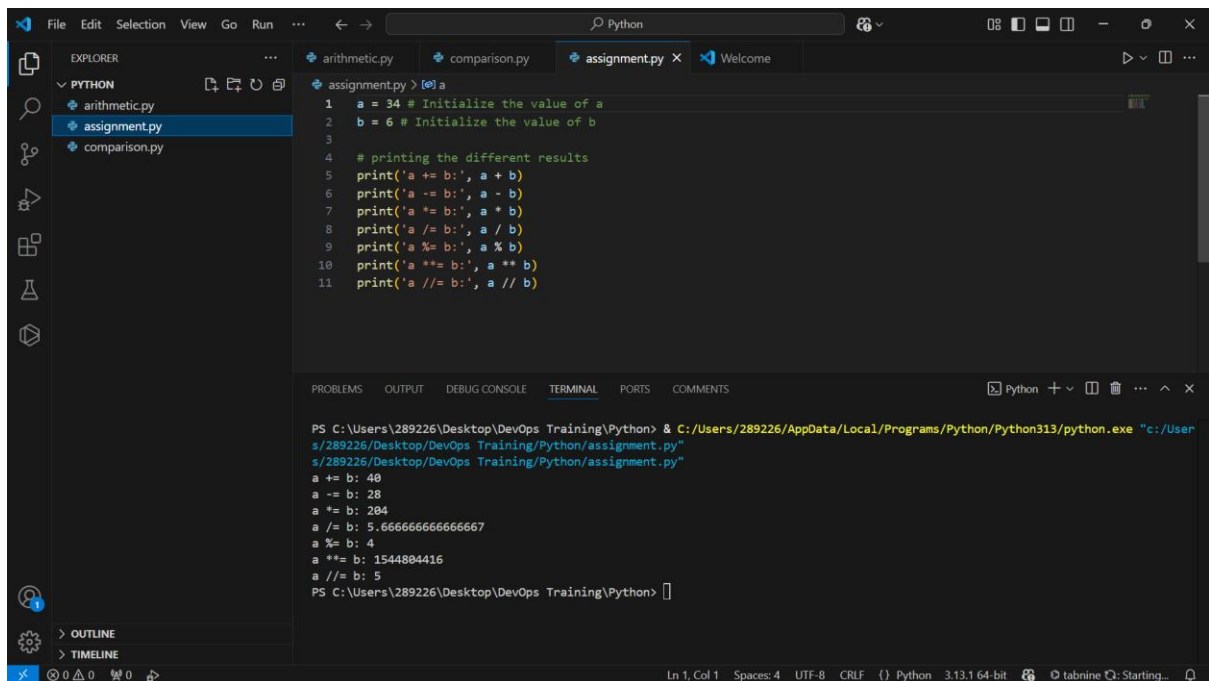


The screenshot shows a Visual Studio Code editor with a file named `comparison.py` open. The code defines two variables, `a = 46` and `b = 4`, and prints the results of various comparison operations. The terminal output shows the execution of the script, displaying the boolean results for each comparison.

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

```
s/289226/Desktop/DevOps Training/Python/comparison.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\289226\Desktop\DevOps Training\Python>
```

ASSIGNMENT OPERATORS:

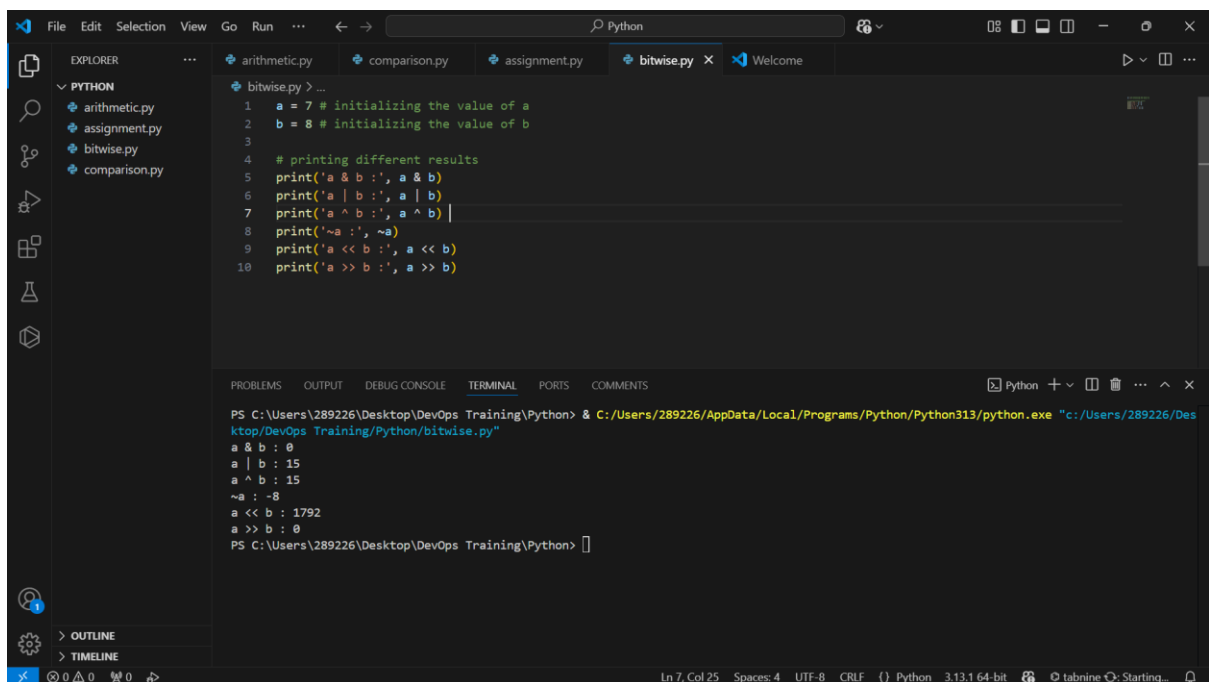


The screenshot shows a Visual Studio Code editor with a Python file named `assignment.py`. The code defines two variables, `a` and `b`, and demonstrates various assignment operators. The terminal output shows the results of these operations.

```
1 a = 34 # Initialize the value of a
2 b = 6 # Initialize the value of b
3
4 # printing the different results
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)
```

```
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/289226/Desktop/DevOps Training/Python/assignment.py"
a += b: 40
a -= b: 28
a *= b: 204
a /= b: 5.666666666666667
a %= b: 4
a **= b: 1544804416
a //= b: 5
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

BITWISE OPERATORS:

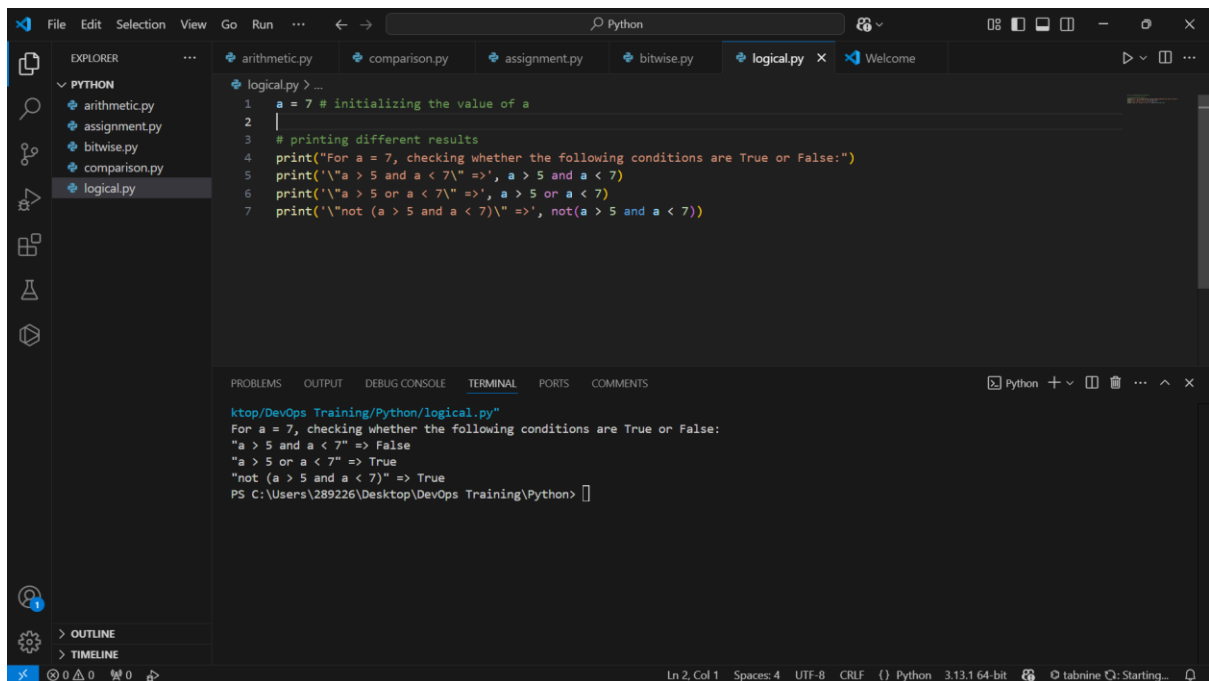


The screenshot shows a Visual Studio Code editor with a Python file named `bitwise.py`. The code defines two variables, `a` and `b`, and demonstrates various bitwise operators. The terminal output shows the results of these operations.

```
1 a = 7 # initializing the value of a
2 b = 8 # initializing the value of b
3
4 # printing different results
5 print('a & b:', a & b)
6 print('a | b:', a | b)
7 print('a ^ b:', a ^ b)
8 print('~a:', ~a)
9 print('a << b:', a << b)
10 print('a >> b:', a >> b)
```

```
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/289226/Desktop/DevOps Training/Python/bitwise.py"
a & b: 0
a | b: 15
a ^ b: 15
~a: -8
a << b: 1792
a >> b: 0
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

LOGICAL OPERATORS:



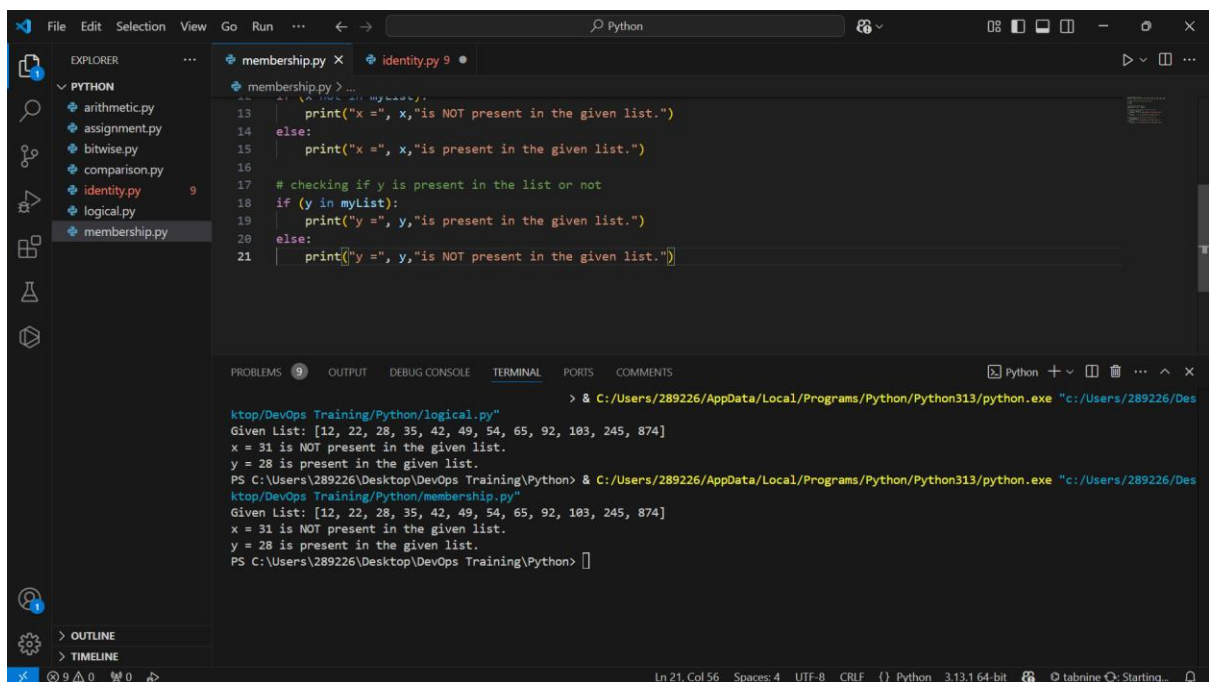
The screenshot shows the Visual Studio Code editor with a Python file named `logical.py`. The code defines a variable `a = 7` and prints the results of several logical conditions. The terminal output shows the execution of these conditions.

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

```
Python
+ v
Python 3.13.1 64-bit
tabnine
Starting...
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
ktop/DevOps Training/Python/logical.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\289226\Desktop\DevOps Training\Python>
```

MEMBERSHIP OPERATORS:



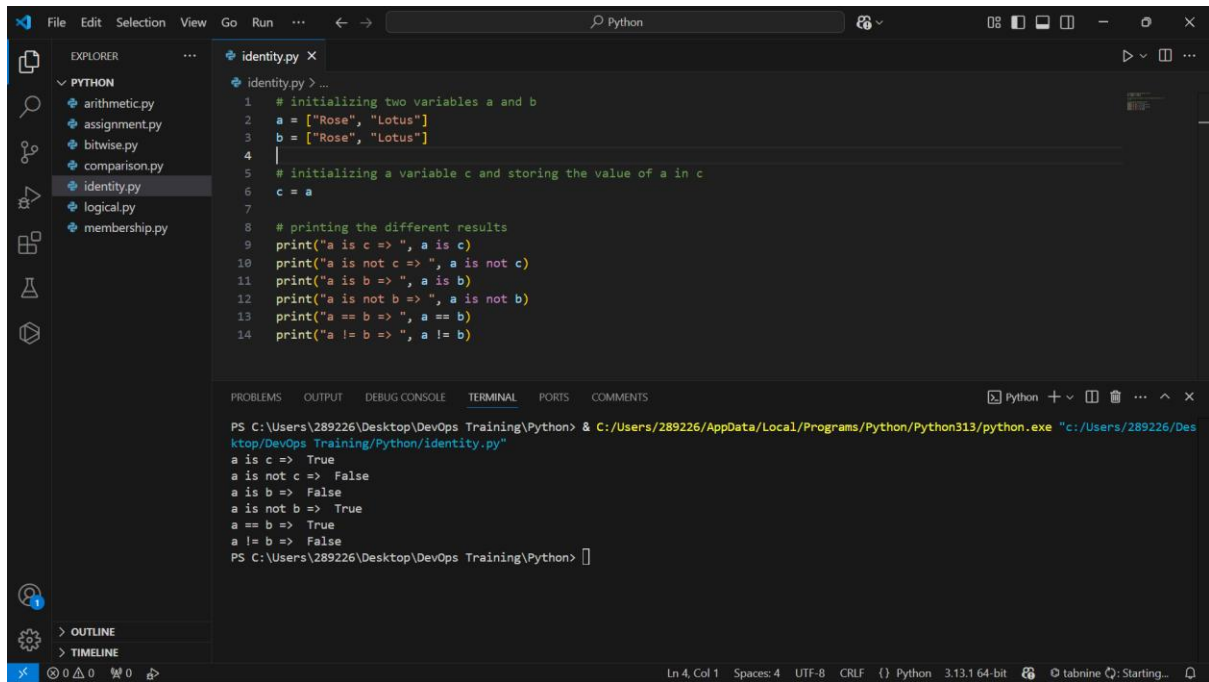
The screenshot shows the Visual Studio Code editor with a Python file named `membership.py`. The code defines a list `mylist` and checks for the presence of elements `x` and `y` using the `in` and `not in` operators. The terminal output shows the execution of these checks.

```
13 print("x =", x,"is NOT present in the given list.")
14 else:
15     print("x =", x,"is present in the given list.")
16
17 # checking if y is present in the list or not
18 if (y in mylist):
19     print("y =", y,"is present in the given list.")
20 else:
21     print("y =", y,"is NOT present in the given list.")
```

```
Python
+ v
Python 3.13.1 64-bit
tabnine
Starting...
```

```
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
> & C:/Users/289226/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/289226/Des
ktop/DevOps Training/Python/logical.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.
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/289226/Des
ktop/DevOps Training/Python/membership.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.
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

IDENTITY OPERATORS:



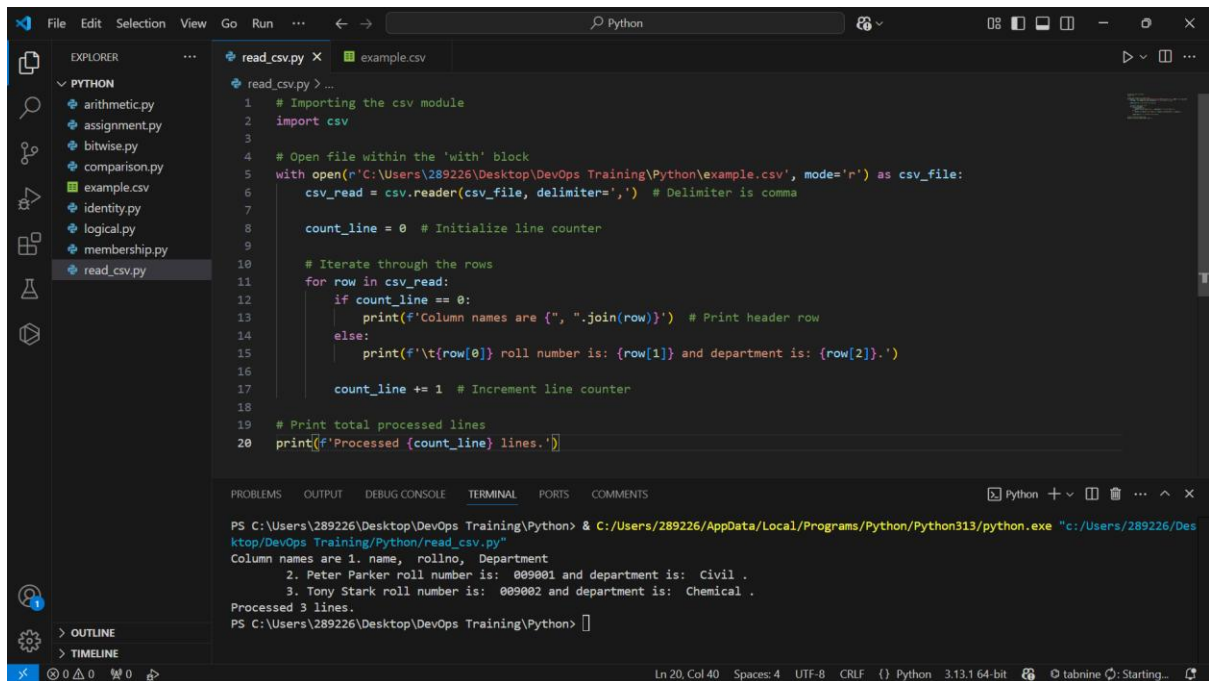
The screenshot shows a Visual Studio Code editor with a Python file named `identity.py`. The file contains a script that demonstrates identity operators (`is`, `is not`, `==`, `!=`) using two lists, `a` and `b`, which both point to the same memory location. A third variable `c` is created as a copy of `a`, pointing to a different memory location.

```
1 # initializing two variables a and b
2 a = ["Rose", "Lotus"]
3 b = ["Rose", "Lotus"]
4
5 # initializing a variable c and storing the value of a in c
6 c = a
7
8 # printing the different results
9 print("a is c => ", a is c)
10 print("a is not c => ", a is not c)
11 print("a is b => ", a is b)
12 print("a is not b => ", a is not b)
13 print("a == b => ", a == b)
14 print("a != b => ", a != b)
```

The terminal output shows the execution of the script, confirming that `a` and `b` are identical (point to the same memory), while `a` and `c` are not identical (point to different memory locations).

```
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/289226/Desktop/DevOps Training/Python/identity.py"
a is c => True
a is not c => False
a is b => True
a is not b => False
a == b => True
a != b => False
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

READ ACSV FILE:

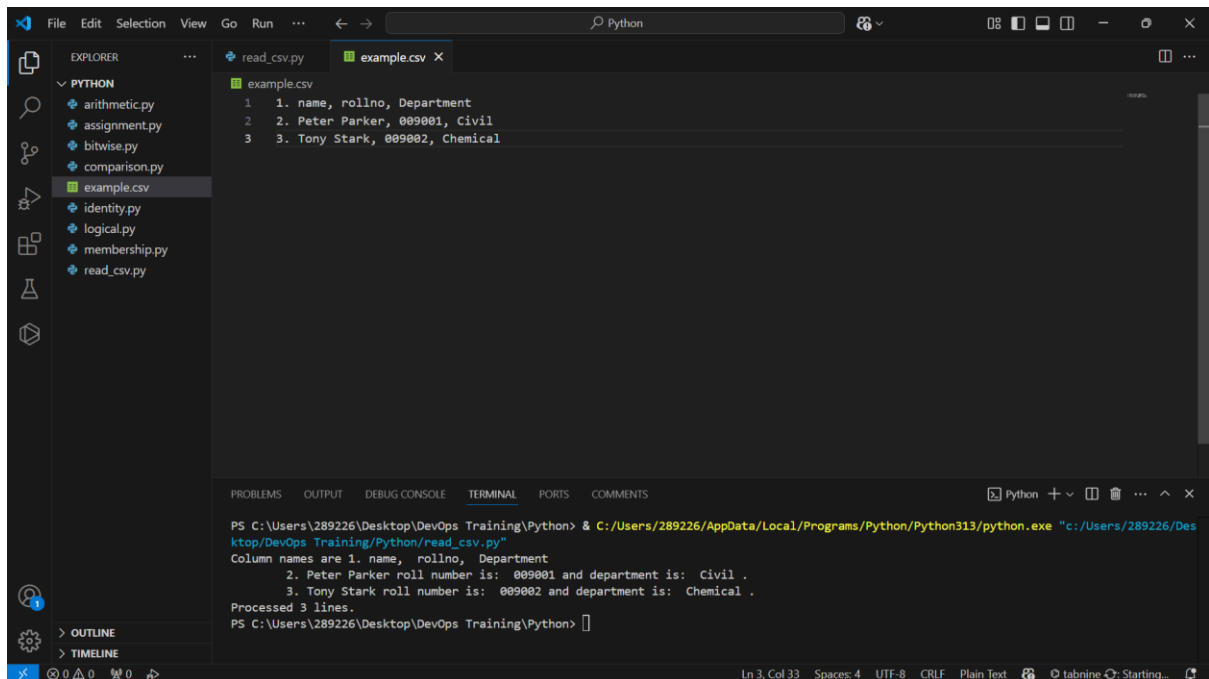


The screenshot shows the Visual Studio Code editor with a Python file named `read_csv.py` open. The file contains a script that reads a CSV file named `example.csv`. The script imports the `csv` module, opens the file, and iterates through its rows. It prints the column names and the roll number and department for each row. The terminal output shows the execution of the script, which successfully reads the CSV file and prints the expected output.

```
1 # Importing the csv module
2 import csv
3
4 # Open file within the 'with' block
5 with open('C:\Users\289226\Desktop\DevOps Training\Python\example.csv', mode='r') as csv_file:
6     csv_read = csv.reader(csv_file, delimiter=',') # Delimiter is comma
7
8     count_line = 0 # Initialize line counter
9
10    # Iterate through the rows
11    for row in csv_read:
12        if count_line == 0:
13            print(f'Column names are {", ".join(row)}') # Print header row
14        else:
15            print(f'\t{row[0]} roll number is: {row[1]} and department is: {row[2]}')
16
17        count_line += 1 # Increment line counter
18
19    # Print total processed lines
20    print(f'Processed {count_line} lines.')
```

Terminal Output:

```
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/289226/Desktop/DevOps Training/Python/read_csv.py"
Column names are 1. name, rollno, Department
2. Peter Parker roll number is: 009001 and department is: Civil .
3. Tony Stark roll number is: 009002 and department is: Chemical .
Processed 3 lines.
PS C:\Users\289226\Desktop\DevOps Training\Python>
```



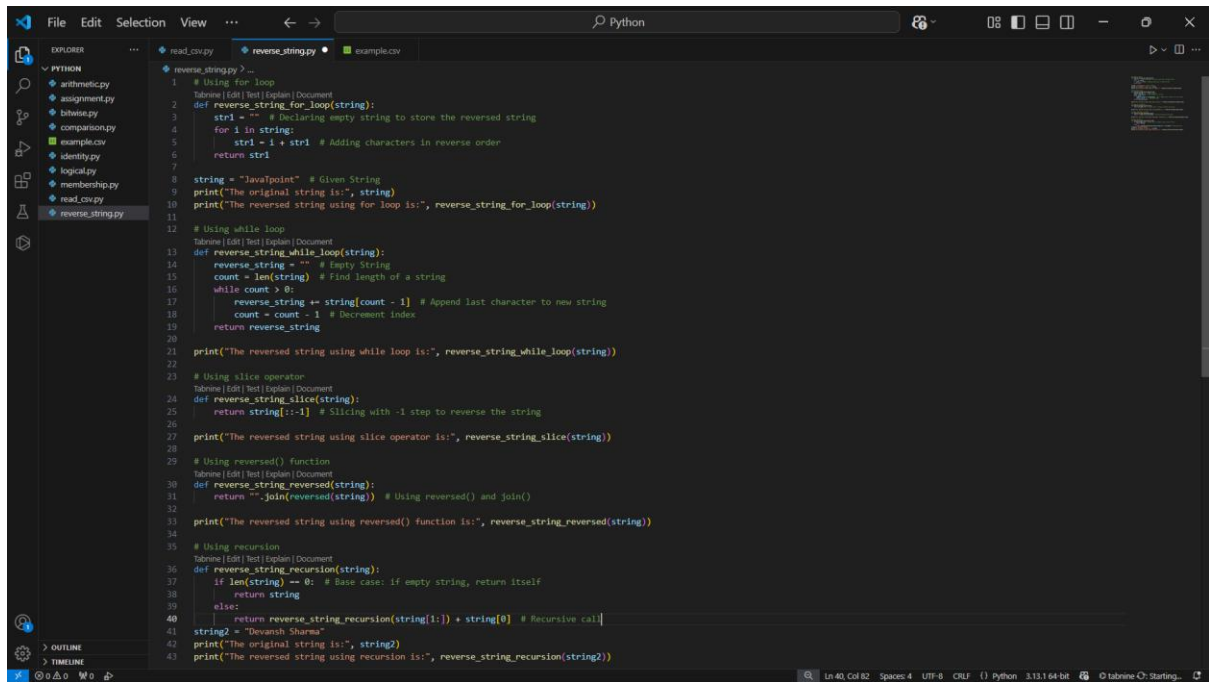
The screenshot shows the Visual Studio Code editor with the `example.csv` file open. The file contains three lines of CSV data. The terminal output shows the execution of the script, which successfully reads the CSV file and prints the expected output.

```
1 1. name, rollno, Department
2 2. Peter Parker, 009001, Civil
3 3. Tony Stark, 009002, Chemical
```

Terminal Output:

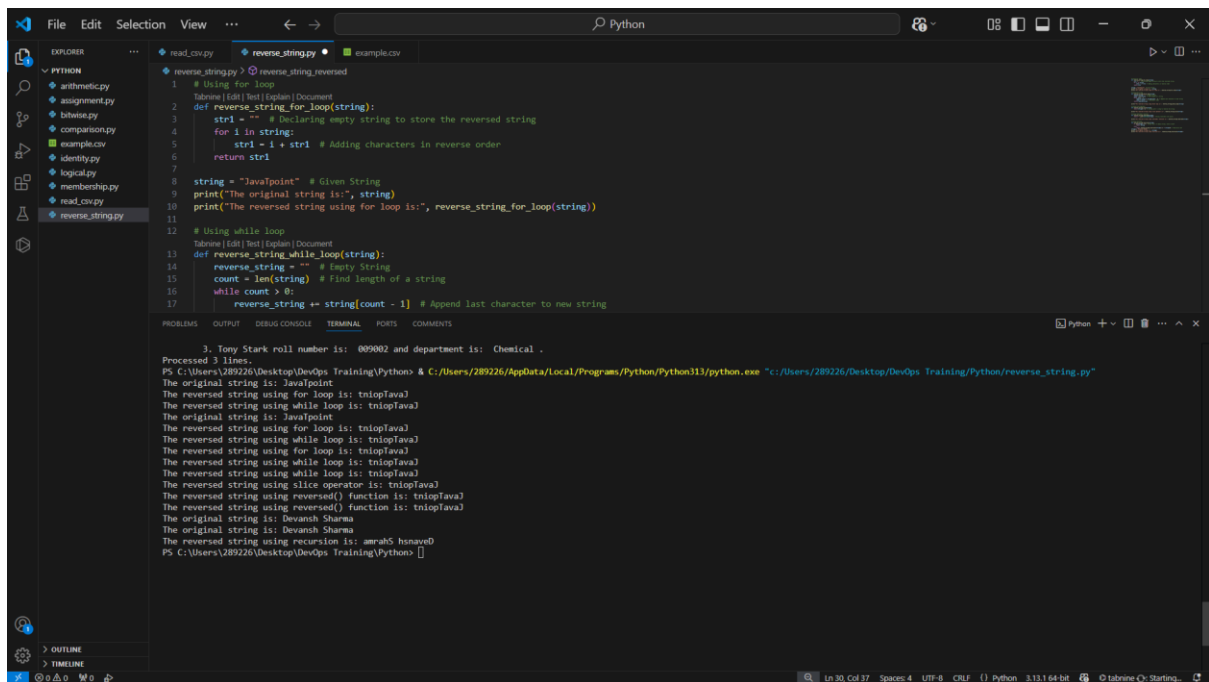
```
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/289226/Desktop/DevOps Training/Python/read_csv.py"
Column names are 1. name, rollno, Department
2. Peter Parker roll number is: 009001 and department is: Civil .
3. Tony Stark roll number is: 009002 and department is: Chemical .
Processed 3 lines.
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

REVERSE STRING:



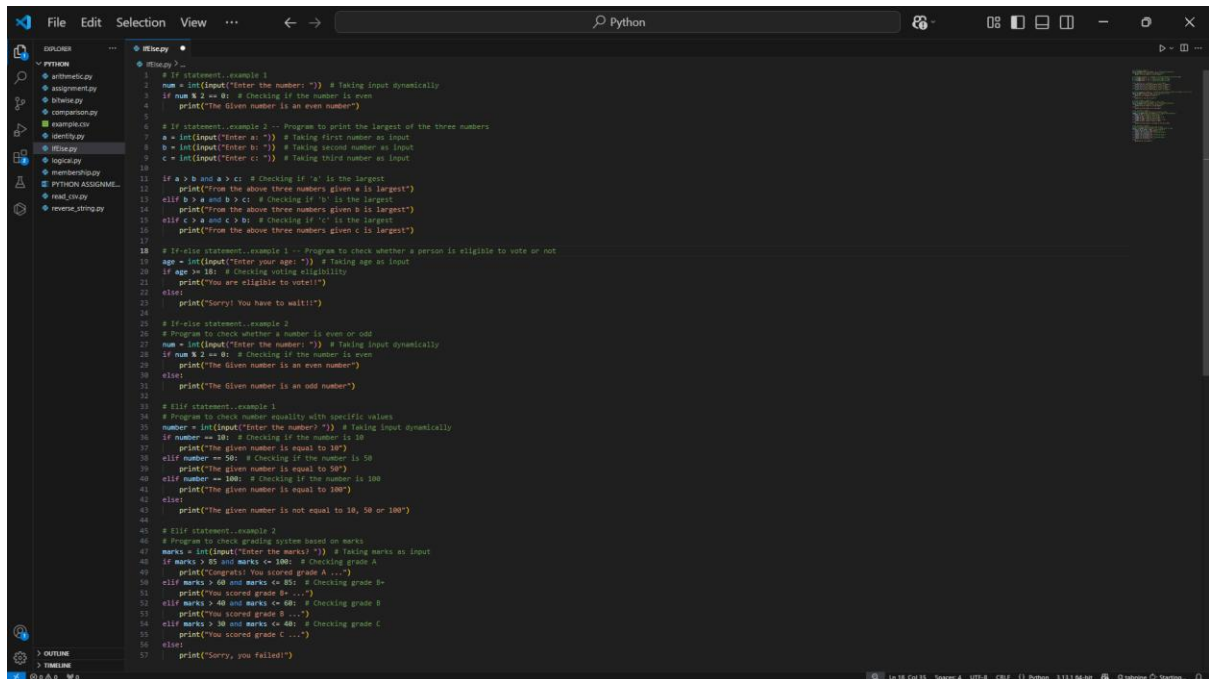
```
1 # Using for loop
2 def reverse_string_for_loop(string):
3     str1 = "" # Declaring empty string to store the reversed string
4     for i in string:
5         str1 = i + str1 # Adding characters in reverse order
6     return str1
7
8 string = "JavaPoint" # Given String
9 print("The original string is:", string)
10 print("The reversed string using for loop is:", reverse_string_for_loop(string))
11
12 # Using while loop
13 def reverse_string_while_loop(string):
14     reverse_string = "" # Empty String
15     count = len(string) # Find length of a string
16     while count > 0:
17         reverse_string += string[count - 1] # Append last character to new string
18         count = count - 1 # Decrement index
19     return reverse_string
20
21 print("The reversed string using while loop is:", reverse_string_while_loop(string))
22
23 # Using slice operator
24 def reverse_string_slice(string):
25     return string[::-1] # Slicing with -1 step to reverse the string
26
27 print("The reversed string using slice operator is:", reverse_string_slice(string))
28
29 # Using reversed() function
30 def reverse_string_reversed(string):
31     return "".join(reversed(string)) # Using reversed() and join()
32
33 print("The reversed string using reversed() function is:", reverse_string_reversed(string))
34
35 # Using recursion
36 def reverse_string_recursion(string):
37     if len(string) == 0: # Base case: if empty string, return itself
38         return string
39     else:
40         return reverse_string_recursion(string[1:]) + string[0] # Recursive call
41
42 string2 = "Devansh Sharma"
43 print("The original string is:", string2)
44 print("The reversed string using recursion is:", reverse_string_recursion(string2))
```

OUTPUT:



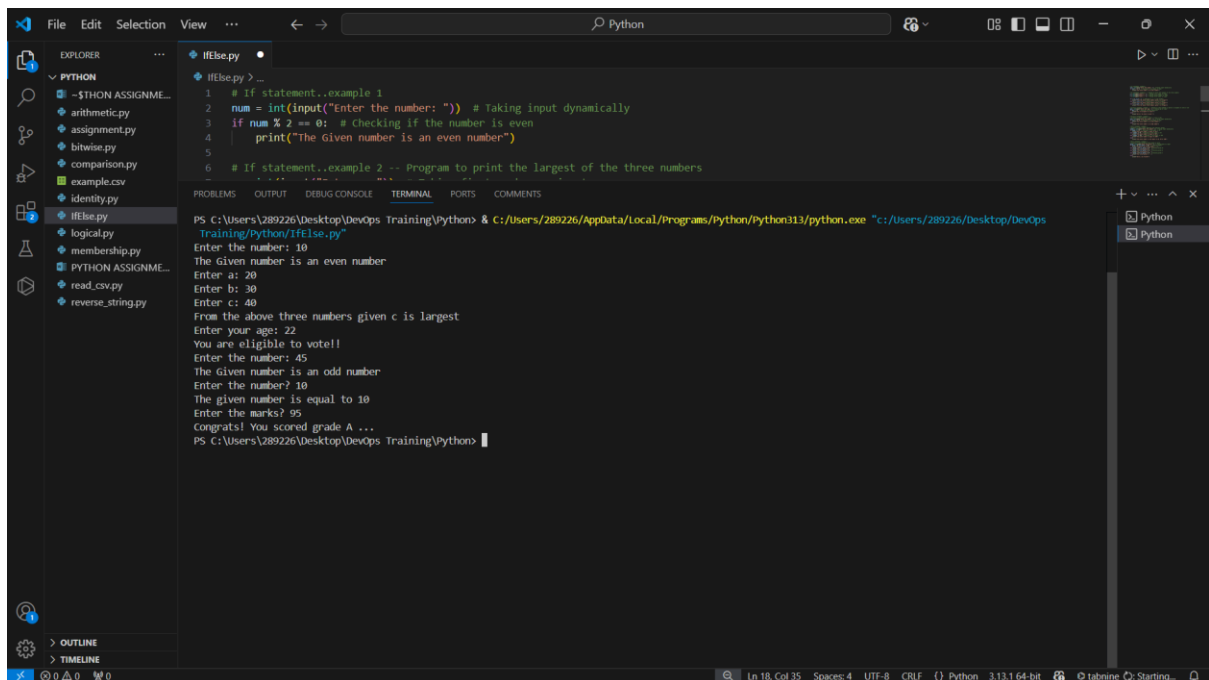
```
3. Tony Stark roll number is: 009002 and department is: Chemical .
Processed 3 lines.
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\289226\Desktop\DevOps Training\Python\reverse_string.py"
The original string is: JavaPoint
The reversed string using for loop is: tnioPlavaJ
The original string is: JavaPoint
The reversed string using while loop is: tnioPlavaJ
The original string is: JavaPoint
The reversed string using slice operator is: tnioPlavaJ
The original string is: JavaPoint
The reversed string using reversed() function is: tnioPlavaJ
The original string is: Devansh Sharma
The reversed string using recursion is: amrahS hnsaeD
PS C:\Users\289226\Desktop\DevOps Training\Python> []
```

IF-ELSE STATEMENTS:



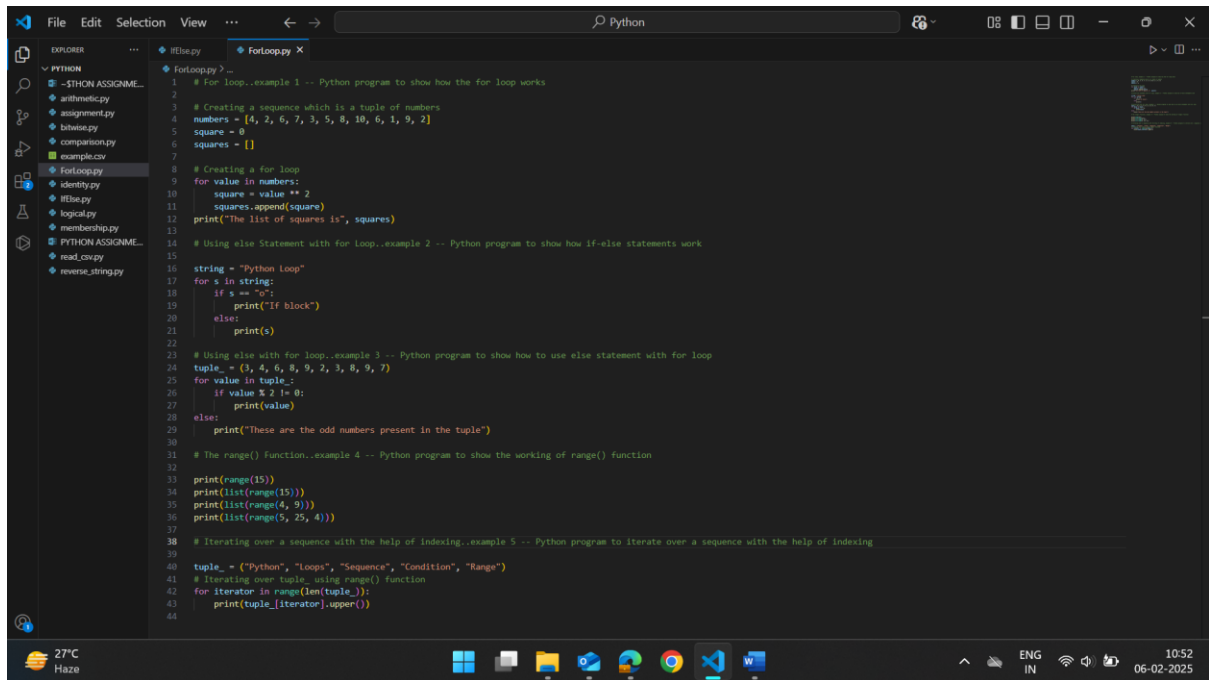
```
1 # If statement..example 1
2 num = int(input("Enter the number: ")) # Taking input dynamically
3 if num % 2 == 0: # Checking if the number is even
4     print("The Given number is an even number")
5
6 # If statement..example 2 -- Program to print the largest of the three numbers
7 a = int(input("Enter a: ")) # Taking first number as input
8 b = int(input("Enter b: ")) # Taking second number as input
9 c = int(input("Enter c: ")) # Taking third number as input
10
11 if a > b and a > c: # Checking if 'a' is the largest
12     print("From the above three numbers given a is largest")
13 elif b > a and b > c: # Checking if 'b' is the largest
14     print("From the above three numbers given b is largest")
15 elif c > a and c > b: # Checking if 'c' is the largest
16     print("From the above three numbers given c is largest")
17
18 # If-else statement..example 1 -- Program to check whether a person is eligible to vote or not
19 age = int(input("Enter your age: ")) # Taking age as input
20 if age >= 18: # Checking voting eligibility
21     print("You are eligible to vote!!")
22 else:
23     print("Sorry! You have to wait!!")
24
25 # If-else statement..example 2
26 # Program to check whether a number is even or odd
27 num = int(input("Enter the number: ")) # Taking input dynamically
28 if num % 2 == 0: # Checking if the number is even
29     print("The Given number is an even number")
30 else:
31     print("The Given number is an odd number")
32
33 # If statement..example 1
34 # Program to check number equality with specific values
35 number = int(input("Enter the number: ")) # Taking input dynamically
36 if number == 10: # Checking if the number is 10
37     print("The given number is equal to 10")
38 elif number == 50: # Checking if the number is 50
39     print("The given number is equal to 50")
40 elif number == 100: # Checking if the number is 100
41     print("The given number is equal to 100")
42 else:
43     print("The given number is not equal to 10, 50 or 100")
44
45 # If statement..example 2
46 # Program to check grading system based on marks
47 marks = int(input("Enter the marks: ")) # Taking marks as input
48 if marks >= 85 and marks <= 100: # Checking grade A
49     print("Congrats! You scored grade A ...")
50 elif marks > 60 and marks <= 85: # Checking grade B
51     print("You scored grade B ...")
52 elif marks > 40 and marks <= 60: # Checking grade C
53     print("You scored grade C ...")
54 elif marks > 30 and marks <= 40: # Checking grade D
55     print("You scored grade D ...")
56 else:
57     print("Sorry, you failed!")
```

OUTPUT:



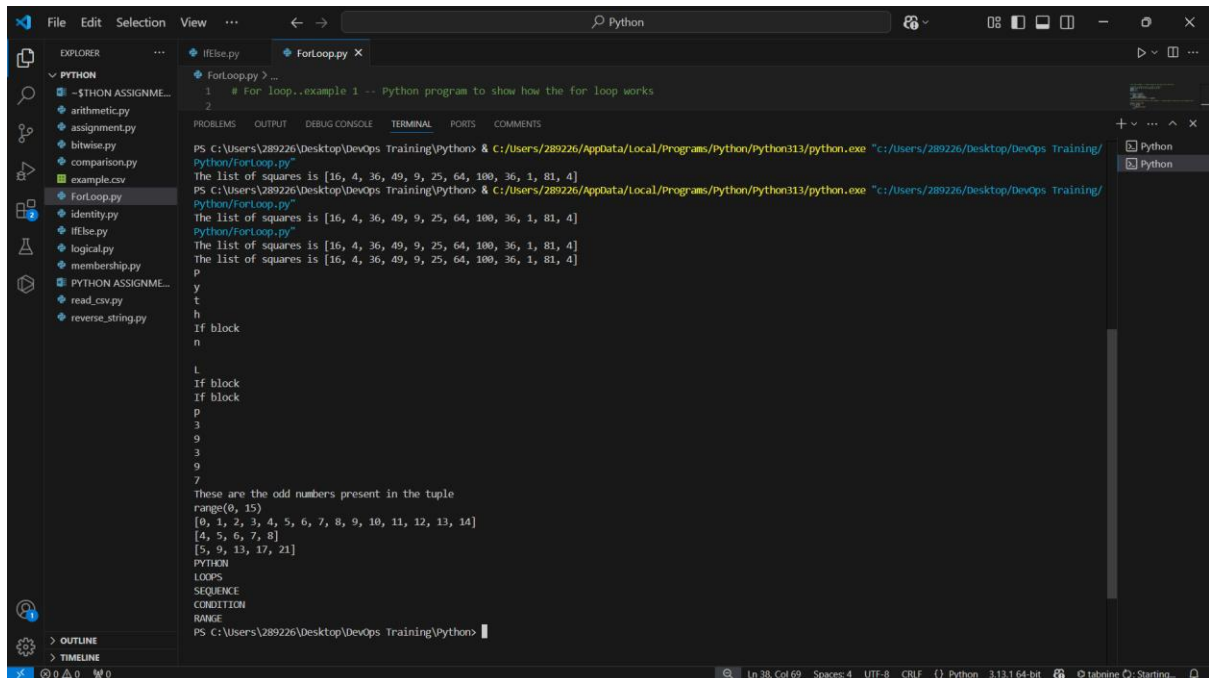
```
PS C:\Users\289226\Desktop\DevOps Training\Python> & c:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\289226\Desktop\DevOps Training\Python\IfElse.py"
Enter the number: 10
The Given number is an even number
Enter a: 20
Enter b: 30
Enter c: 40
From the above three numbers given c is largest
Enter your age: 22
You are eligible to vote!!
Enter the number: 45
The Given number is an odd number
Enter the number: 10
The given number is equal to 10
Enter the marks: 95
Congrats! You scored grade A ...
PS C:\Users\289226\Desktop\DevOps Training\Python>
```


LOOP STATEMENTS: FOR LOOP



```
1 # For loop..example 1 -- Python program to show how the for loop works
2
3 # Creating a sequence which is a tuple of numbers
4 numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]
5 square = 0
6 squares = []
7
8 # Creating a for loop
9 for value in numbers:
10     square = value ** 2
11     squares.append(square)
12 print("The list of squares is", squares)
13
14 # Using else Statement with for loop..example 2 -- Python program to show how if-else statements work
15
16 string = "Python Loop"
17 for s in string:
18     if s == "o":
19         print("If block")
20     else:
21         print(s)
22
23 # Using else with for loop..example 3 -- Python program to show how to use else statement with for loop
24 tuple = (0, 4, 6, 9, 2, 3, 8, 9, 7)
25 for value in tuple:
26     if value % 2 != 0:
27         print(value)
28 else:
29     print("These are the odd numbers present in the tuple")
30
31 # The range() Function..example 4 -- Python program to show the working of range() function
32
33 print(range(15))
34 print(list(range(15)))
35 print(list(range(4, 9)))
36 print(list(range(5, 25, 4)))
37
38 # Iterating over a sequence with the help of indexing..example 5 -- Python program to iterate over a sequence with the help of indexing
39
40 tuple = ("Python", "Loops", "Sequence", "Condition", "Range")
41 # Iterating over tuple using range() function
42 for iterator in range(len(tuple)):
43     print(tuple[iterator].upper())
44
```

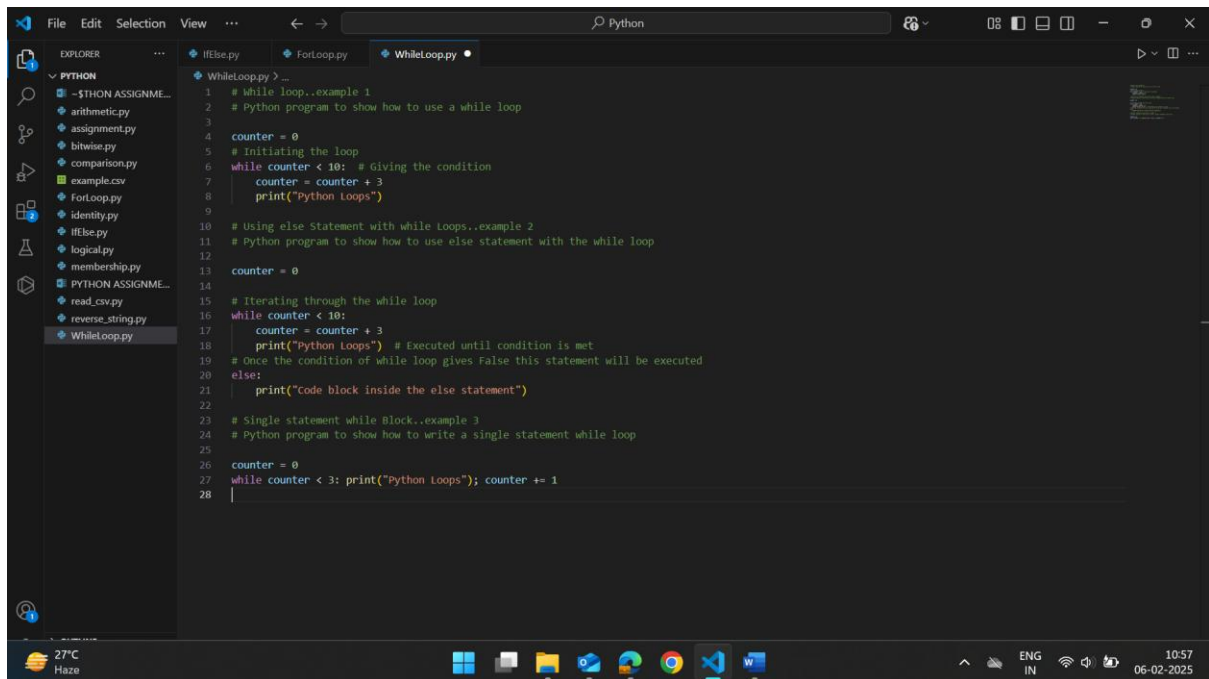
OUTPUT:



```
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\289226\Desktop\DevOps Training\Python\ForLoop.py"
The list of squares is [16, 4, 36, 49, 9, 25, 64, 100, 36, 1, 81, 4]
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\289226\Desktop\DevOps Training\Python\ForLoop.py"
The list of squares is [16, 4, 36, 49, 9, 25, 64, 100, 36, 1, 81, 4]
Python\ForLoop.py
The list of squares is [16, 4, 36, 49, 9, 25, 64, 100, 36, 1, 81, 4]
The list of squares is [16, 4, 36, 49, 9, 25, 64, 100, 36, 1, 81, 4]
p
y
t
h
If block
n

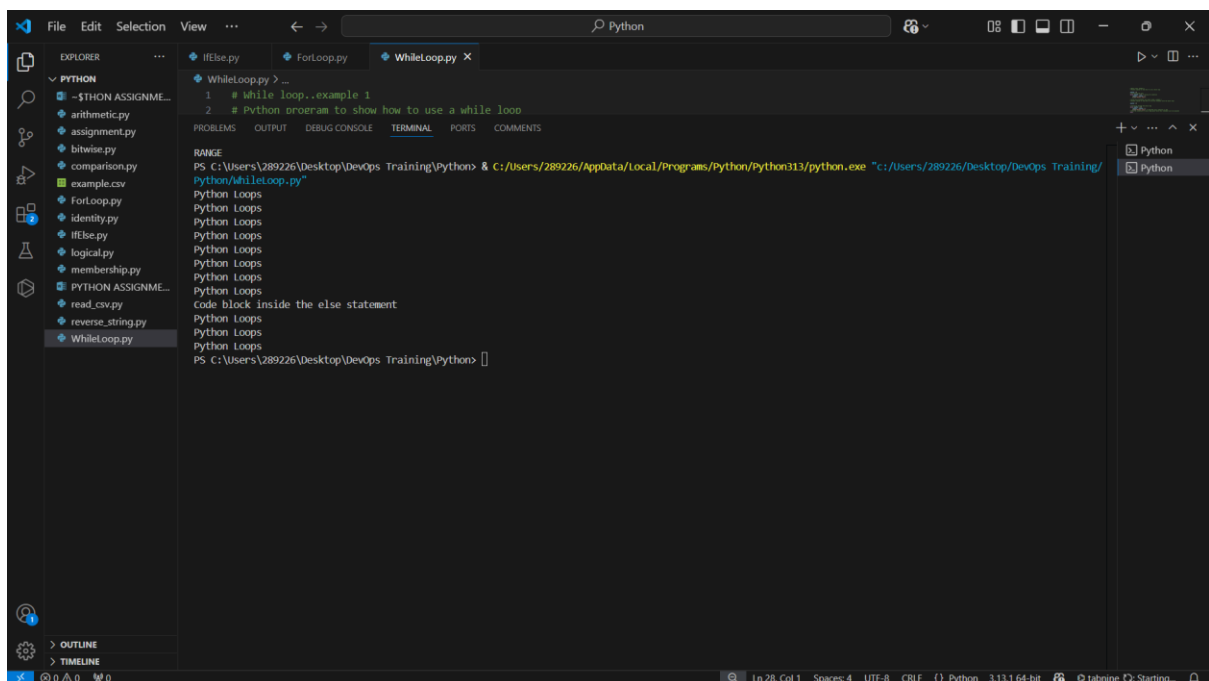
L
If block
If block
p
3
9
3
9
7
These are the odd numbers present in the tuple
range(0, 15)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[4, 5, 6, 7, 8]
[5, 9, 13, 17, 21]
PYTHON
LOOPS
SEQUENCE
CONDITION
RANGE
PS C:\Users\289226\Desktop\DevOps Training\Python>
```


LOOP STATEMENTS: WHILE LOOP



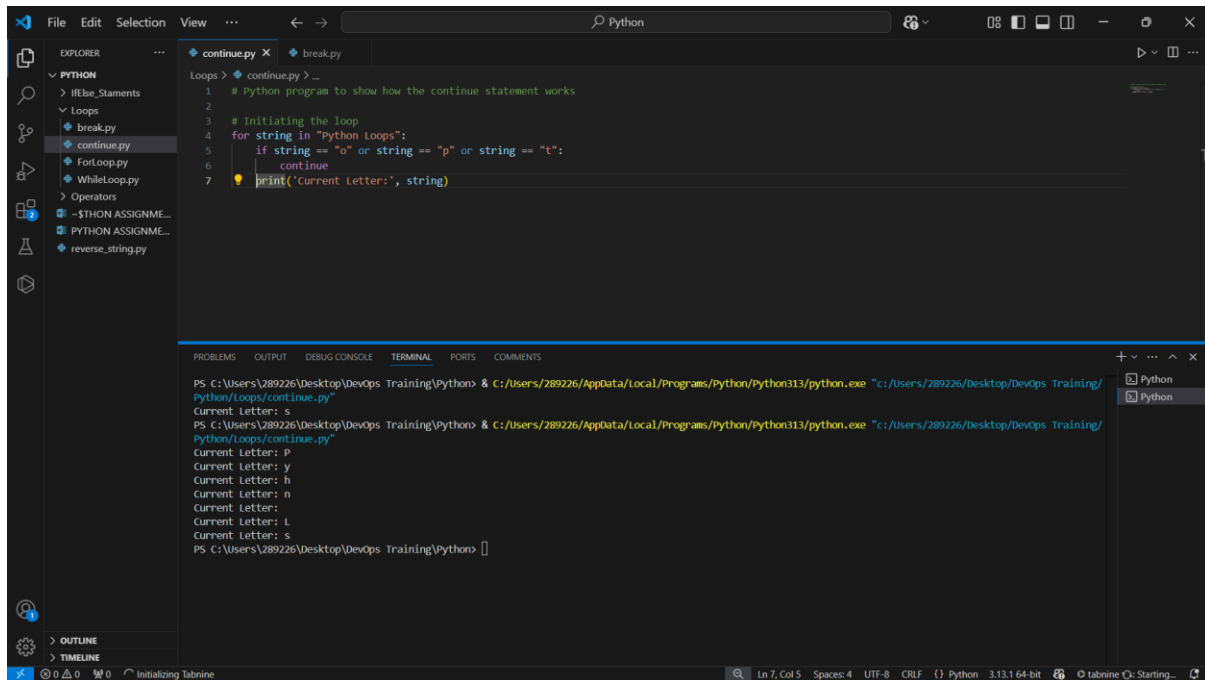
```
1 # While loop..example 1
2 # Python program to show how to use a while loop
3
4 counter = 0
5 # Initiating the loop
6 while counter < 10: # Giving the condition
7     counter = counter + 3
8     print("Python Loops")
9
10 # Using else Statement with while Loops..example 2
11 # Python program to show how to use else statement with the while loop
12
13 counter = 0
14
15 # Iterating through the while loop
16 while counter < 10:
17     counter = counter + 3
18     print("Python Loops") # Executed until condition is met
19 # Once the condition of while loop gives False this statement will be executed
20 else:
21     print("code block inside the else statement")
22
23 # Single statement while Block..example 3
24 # Python program to show how to write a single statement while loop
25
26 counter = 0
27 while counter < 3: print("Python Loops"); counter += 1
28
```

OUTPUT:



```
RANGE
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\289226\Desktop\DevOps Training\Python\WhileLoop.py"
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
code block inside the else statement
Python Loops
Python Loops
Python Loops
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

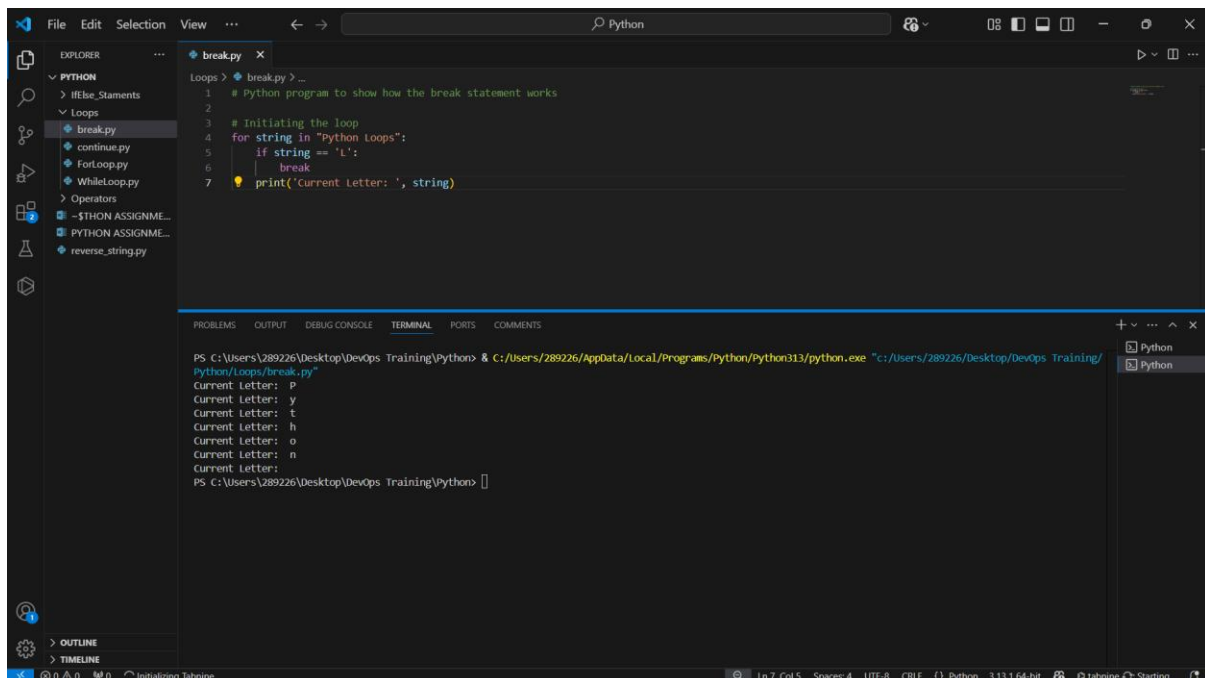
CONTINUE STATEMENT:



```
1 # Python program to show how the continue statement works
2
3 # Initiating the loop
4 for string in "Python Loops":
5     if string == "o" or string == "p" or string == "t":
6         continue
7     print('Current Letter:', string)
```

```
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/289226/Desktop/DevOps Training/Python/Loops/continue.py"
Current Letter: s
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/289226/Desktop/DevOps Training/Python/Loops/continue.py"
Current Letter: P
Current Letter: y
Current Letter: h
Current Letter: n
Current Letter:
Current Letter: L
Current Letter: s
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

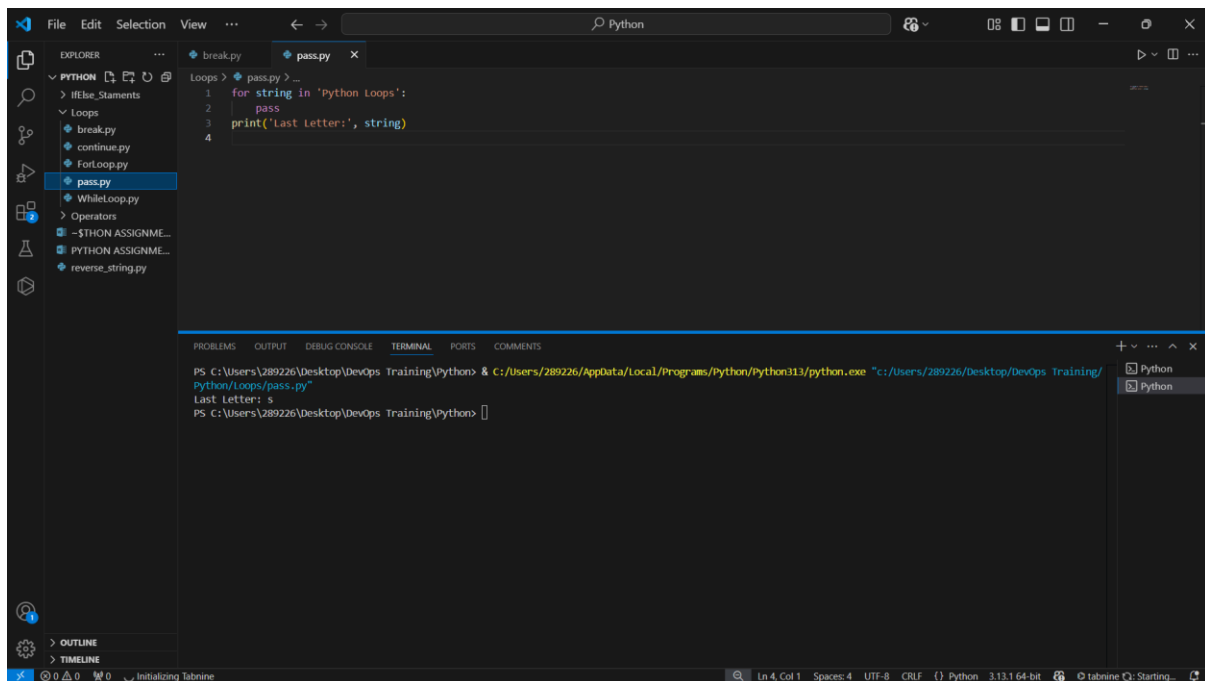
BREAK STATEMENTS:



```
1 # Python program to show how the break statement works
2
3 # Initiating the loop
4 for string in "Python Loops":
5     if string == 'L':
6         break
7     print('Current Letter:', string)
```

```
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/289226/Desktop/DevOps Training/Python/Loops/break.py"
Current Letter: P
Current Letter: y
Current Letter: t
Current Letter: h
Current Letter: o
Current Letter: n
Current Letter:
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

PASS STATEMENT:



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

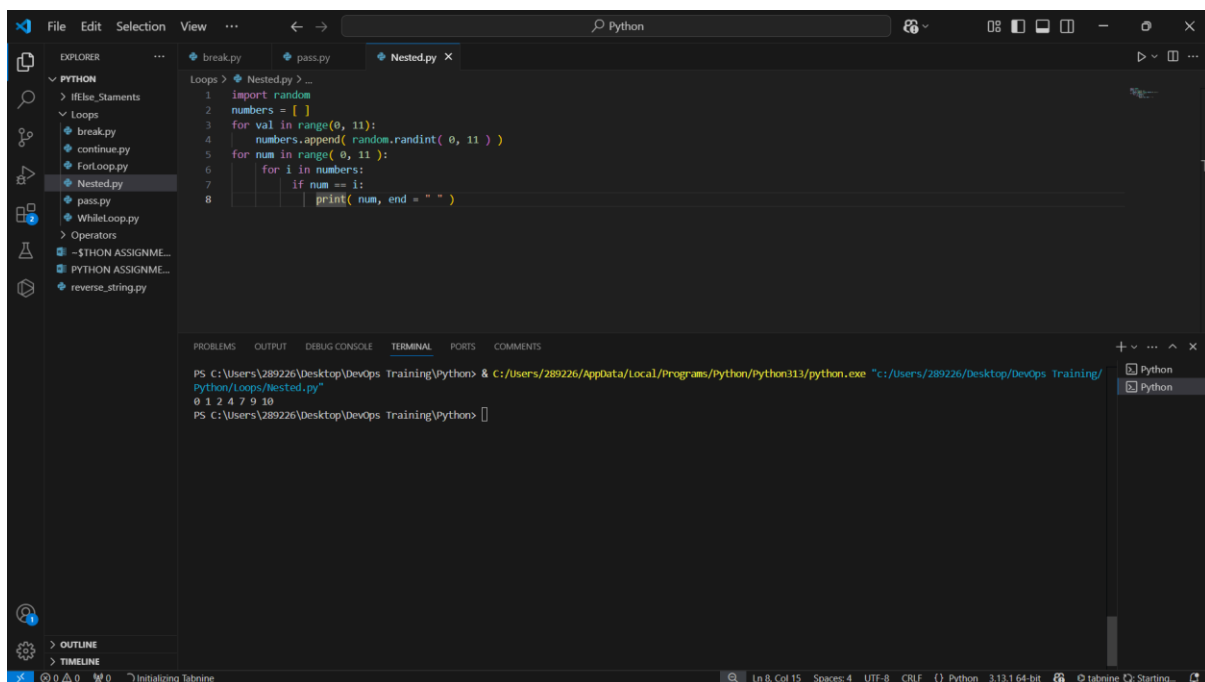
```
1 for string in 'Python Loops':  
2     pass  
3     print('Last Letter:', string)  
4
```

The Explorer sidebar on the left shows a project structure with a `PYTHON` folder containing `ifElse_Statements`, `Loops`, `break.py`, `continue.py`, `ForLoop.py`, `pass.py`, `WhileLoop.py`, `Operators`, `~$THON ASSIGNME...`, `PYTHON ASSIGNME...`, and `reverse_string.py`. The `pass.py` file is selected.

The TERMINAL panel at the bottom shows the command prompt output:

```
PS C:\Users\289226\Desktop\DevOps Training\Python> & c:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\289226\Desktop\DevOps Training\Python\Loops\pass.py"  
Last Letter: s  
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

NESTED LOOPS:



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

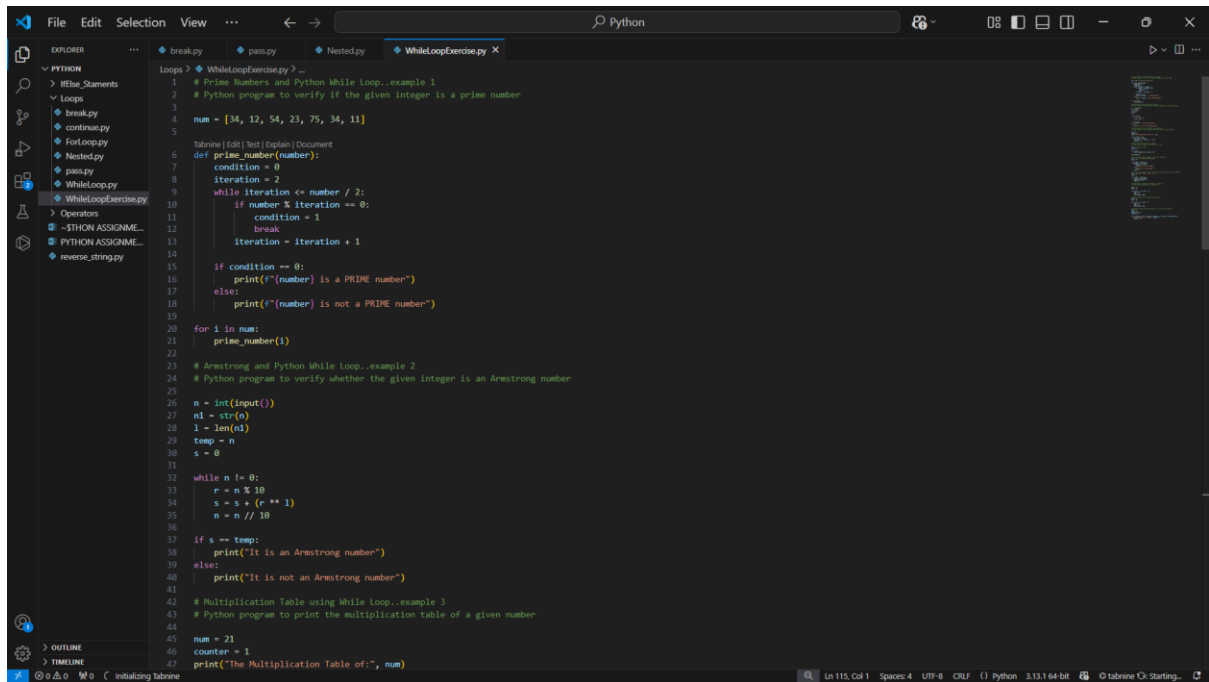
```
1 import random  
2 numbers = []  
3 for val in range(0, 11):  
4     numbers.append( random.randint( 0, 11 ) )  
5 for num in range( 0, 11 ):  
6     for i in numbers:  
7         if num == i:  
8             print( num, end = " " )
```

The Explorer sidebar on the left shows the same project structure as the previous screenshot, with `Nested.py` now selected.

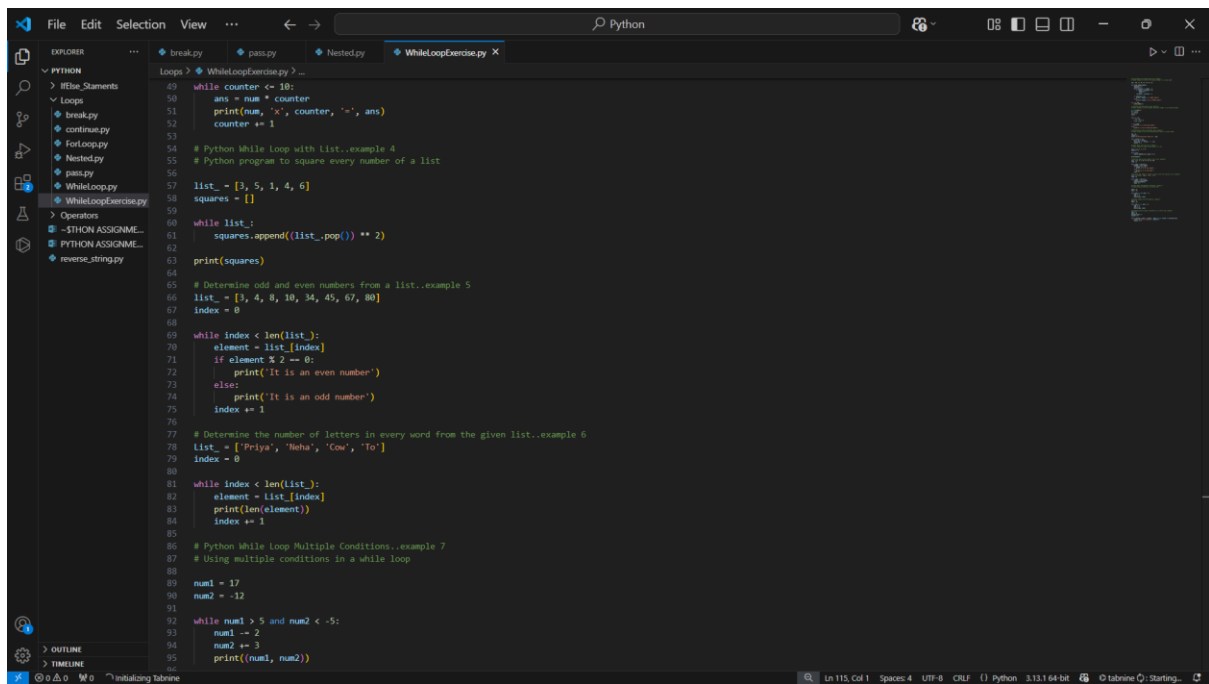
The TERMINAL panel at the bottom shows the command prompt output:

```
PS C:\Users\289226\Desktop\DevOps Training\Python> & c:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\289226\Desktop\DevOps Training\Python\Loops\Nested.py"  
0 1 2 4 7 9 10  
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

EXERCISES ON WHILE LOOP:



```
1 # Prime Numbers and Python While Loop..example 1
2 # Python program to verify if the given integer is a prime number
3
4 num = [34, 12, 54, 23, 75, 34, 11]
5
6 tabnine [Edit] [Test] [Explain] [Document]
7 def prime_number(number):
8     condition = 0
9     iteration = 2
10    while iteration <= number / 2:
11        if number % iteration == 0:
12            condition = 1
13            break
14        iteration = iteration + 1
15
16    if condition == 0:
17        print(f"{number} is a PRIME number")
18    else:
19        print(f"{number} is not a PRIME number")
20
21    for i in num:
22        prime_number(i)
23
24 # Armstrong and Python While Loop..example 2
25 # Python program to verify whether the given integer is an Armstrong number
26
27 n = int(input())
28 n1 = str(n)
29 l = len(n1)
30 temp = n
31 s = 0
32
33 while n != 0:
34     r = n % 10
35     s = s + (r ** l)
36     n = n // 10
37
38 if s == temp:
39     print("It is an Armstrong number")
40 else:
41     print("It is not an Armstrong number")
42
43 # Multiplication Table using While Loop..example 3
44 # Python program to print the multiplication table of a given number
45
46 num = 21
47 counter = 1
48 print("The Multiplication Table of:", num)
```



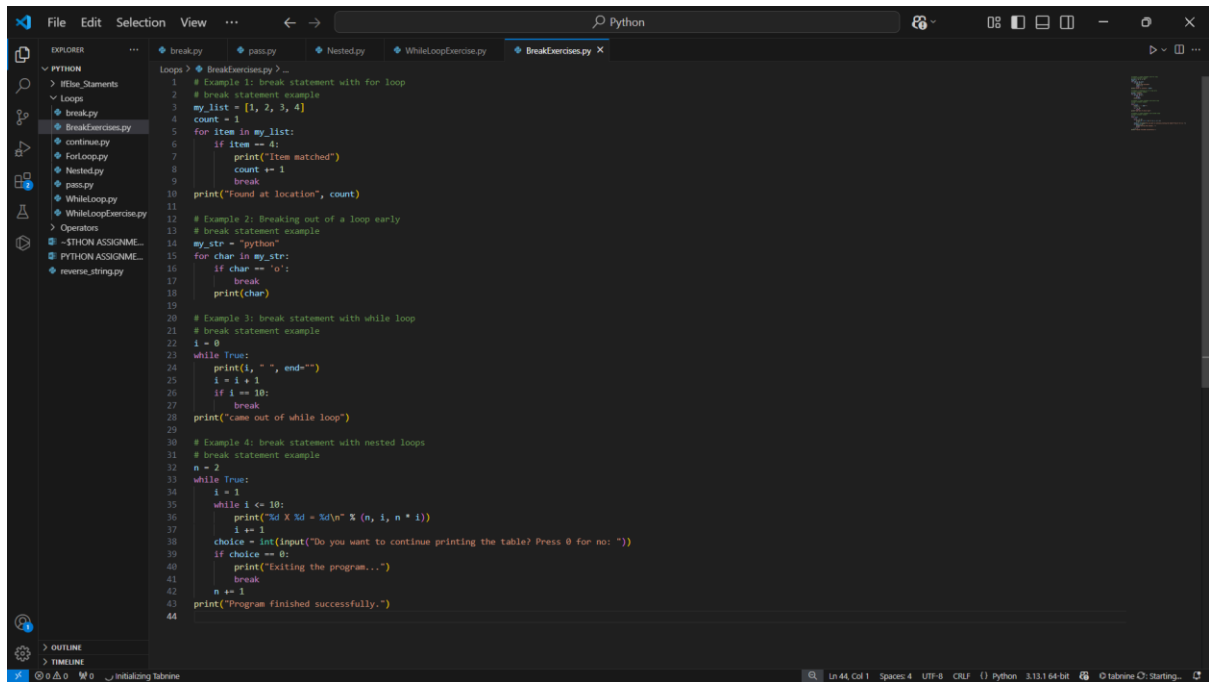
```
49 while counter <= 10:
50     ans = num * counter
51     print(num, 'x', counter, '=', ans)
52     counter += 1
53
54 # Python While Loop with List..example 4
55 # Python program to square every number of a list
56
57 list_ = [3, 5, 1, 4, 6]
58 squares = []
59
60 while list_:
61     squares.append((list_.pop()) ** 2)
62
63 print(squares)
64
65 # Determine odd and even numbers from a list..example 5
66 list_ = [1, 4, 8, 10, 34, 45, 67, 80]
67 index = 0
68
69 while index < len(list_):
70     element = list_[index]
71     if element % 2 == 0:
72         print('It is an even number')
73     else:
74         print('It is an odd number')
75     index += 1
76
77 # Determine the number of letters in every word from the given list..example 6
78 list_ = ['Priya', 'Neha', 'Cow', 'To']
79 index = 0
80
81 while index < len(list_):
82     element = list_[index]
83     print(len(element))
84     index += 1
85
86 # Python While Loop Multiple Conditions..example 7
87 # Using multiple conditions in a while loop
88
89 num1 = 17
90 num2 = -12
91
92 while num1 > 5 and num2 < -5:
93     num1 -= 2
94     num2 += 3
95     print((num1, num2))
96
```

```
96  
97 # Another example with OR condition..example 8  
98 num1 = 17  
99 num2 = -12  
100  
101 while num1 > 5 or num2 < -5:  
102     num1 -= 2  
103     num2 += 3  
104     print((num1, num2))  
105  
106 # Grouping multiple logical expressions in a while loop..example 9  
107 num1 = 9  
108 num2 = 14  
109 maximum_value = 4  
110 counter = 0  
111  
112 while (counter < num1 or counter < num2) and not counter >= maximum_value:  
113     print("Number of iterations: (counter)")  
114     counter += 1  
115
```

OUTPUT:

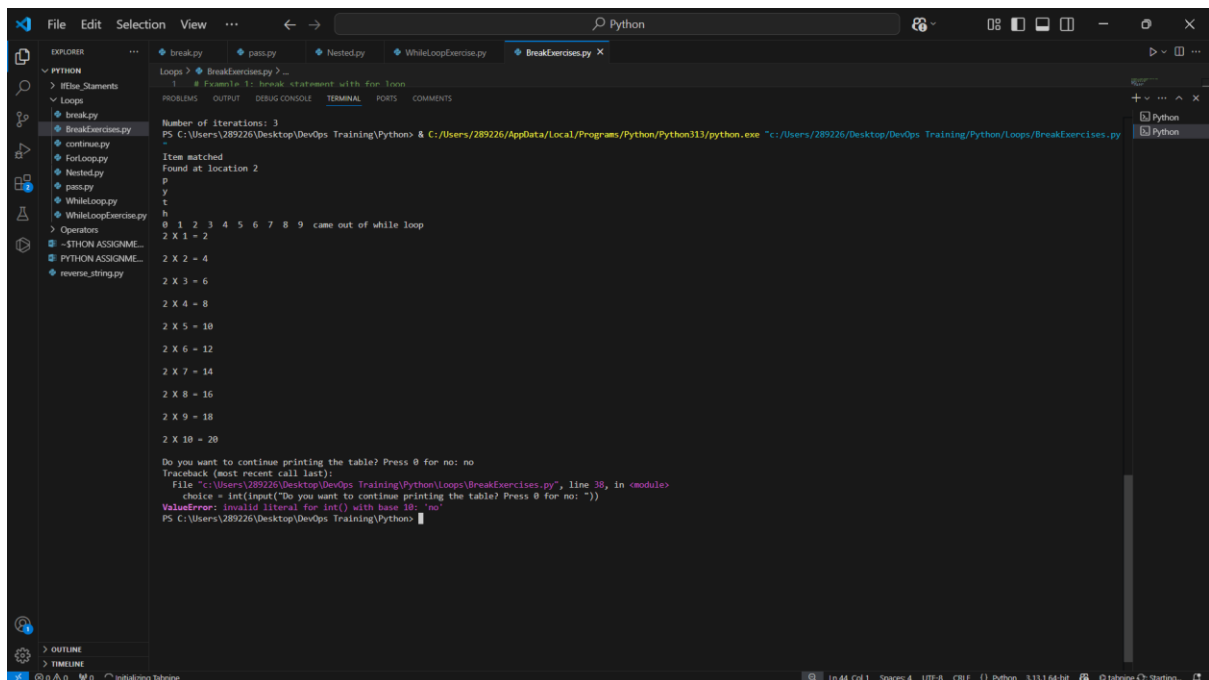
```
75 is not a PRIME number  
34 is not a PRIME number  
11 is a PRIME number  
123  
It is not an Armstrong number  
The Multiplication Table of: 21  
21 x 1 = 21  
21 x 2 = 42  
21 x 3 = 63  
21 x 4 = 84  
21 x 5 = 105  
21 x 6 = 126  
21 x 7 = 147  
21 x 8 = 168  
21 x 9 = 189  
21 x 10 = 210  
[16, 16, 1, 25, 0]  
It is an odd number  
It is an even number  
It is an even number  
It is an even number  
It is an even number  
It is an odd number  
It is an odd number  
It is an even number  
5  
4  
3  
2  
(15, -9)  
(13, -6)  
(11, -3)  
(15, -9)  
(13, -6)  
(11, -3)  
(9, 0)  
(7, 3)  
(5, 6)  
Number of iterations: 0  
Number of iterations: 1  
Number of iterations: 2  
Number of iterations: 3  
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

EXERCISES ON BREAK STATEMENTS:



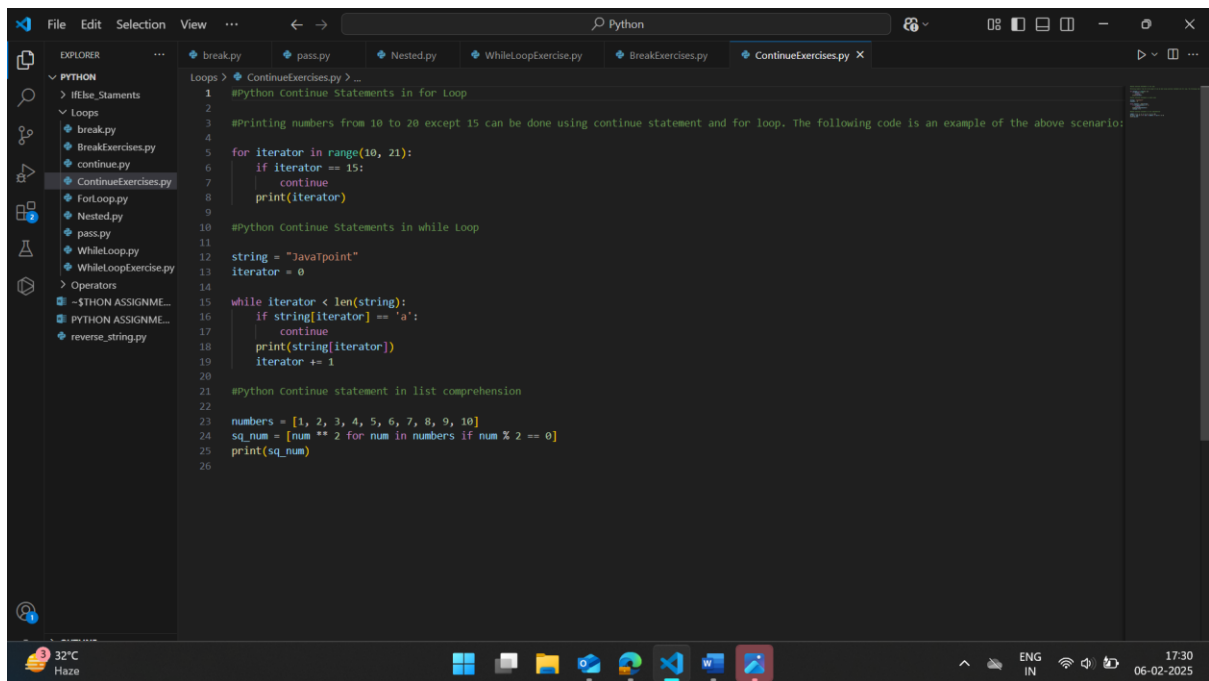
```
1 # Example 1: break statement with for loop
2 # break statement example
3 my_list = [1, 2, 3, 4]
4 count = 1
5 for item in my_list:
6     if item == 4:
7         print("Item matched")
8         count += 1
9         break
10    print("Found at location", count)
11
12 # Example 2: Breaking out of a loop early
13 # break statement example
14 my_str = "python"
15 for char in my_str:
16     if char == 'o':
17         break
18     print(char)
19
20 # Example 3: break statement with while loop
21 # break statement example
22 i = 0
23 while True:
24     print(i, " ", end=" ")
25     i += 1
26     if i == 10:
27         break
28    print("came out of while loop")
29
30 # Example 4: break statement with nested loops
31 # break statement example
32 n = 2
33 while True:
34     i = 1
35     while i <= 10:
36         print("%d X %d = %d" % (n, i, n * i))
37         i += 1
38         choice = int(input("Do you want to continue printing the table? Press 0 for no: "))
39         if choice == 0:
40             print("Exiting the program...")
41             break
42         n += 1
43     print("Program finished successfully.")
44
```

OUTPUT:



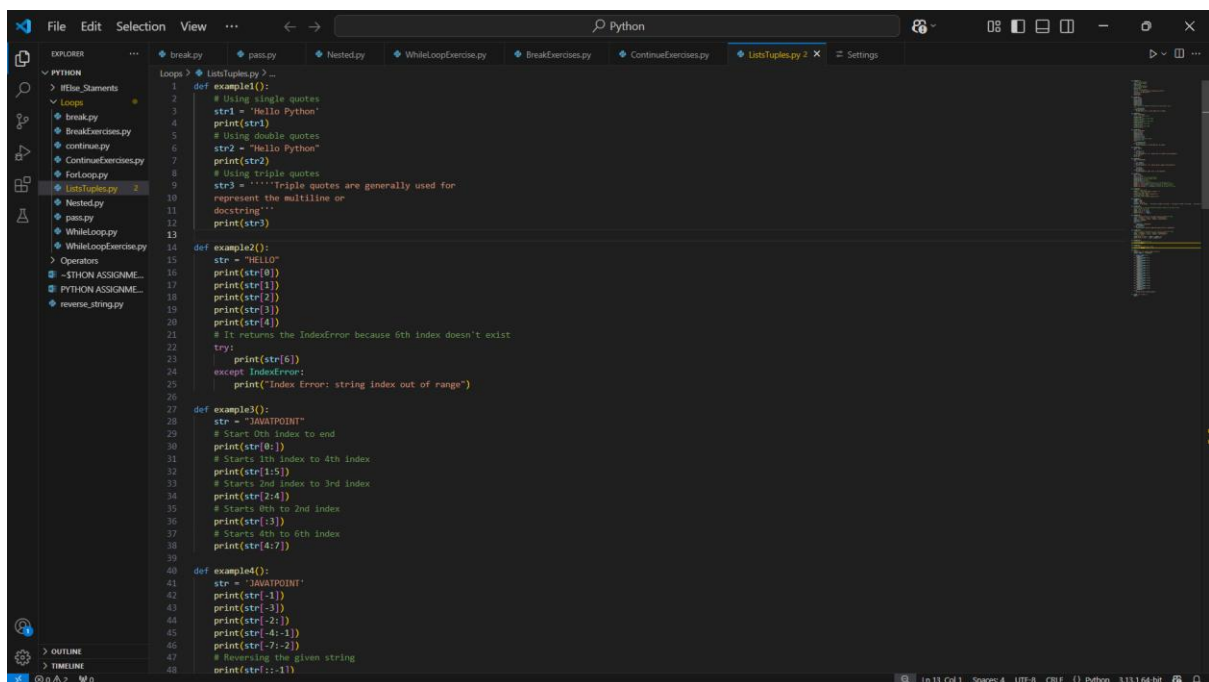
```
Number of iterations: 3
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/289226/Desktop/DevOps Training/Python/Loops/BreakExercises.py"
Item matched
Found at location 2
0
1
2
3
4
5
6
7
8
9
came out of while loop
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20
Do you want to continue printing the table? Press 0 for no: no
Traceback (most recent call last):
  File "c:/Users/289226/Desktop/DevOps Training/Python/Loops/BreakExercises.py", line 38, in <module>
    choice = int(input("Do you want to continue printing the table? Press 0 for no: "))
ValueError: invalid literal for int() with base 10: 'no'
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

CONTINUE STATEMENTS EXERCISES:

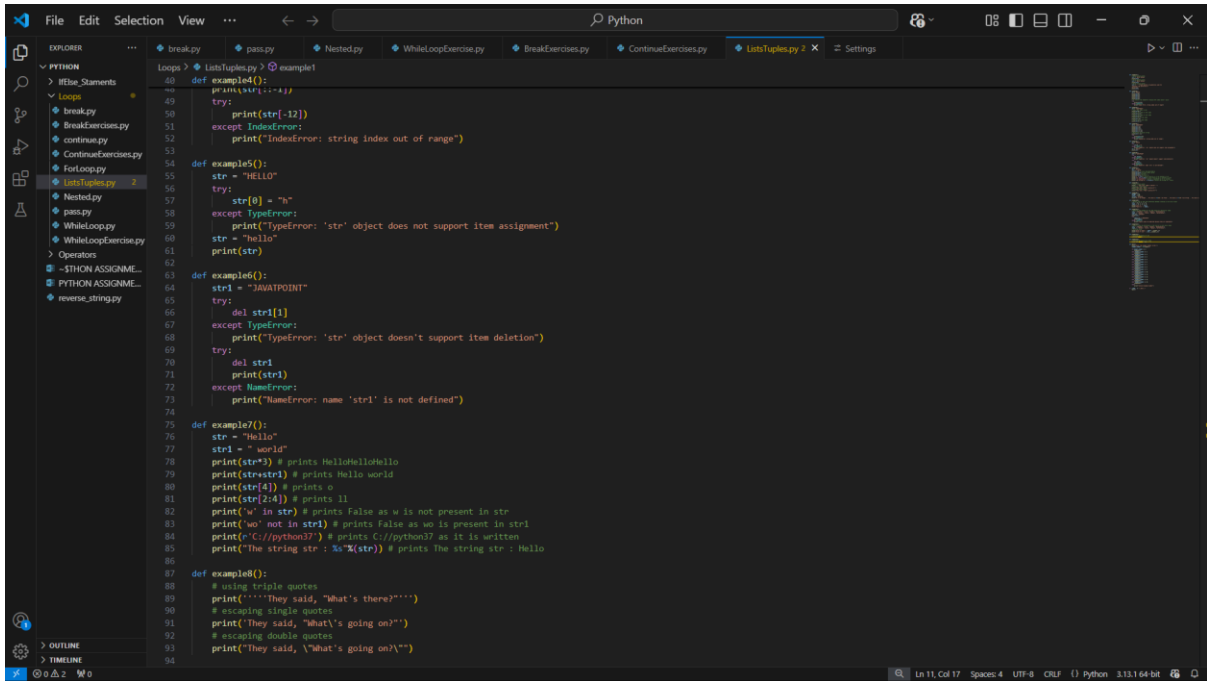


```
1 #Python Continue Statements in for Loop
2
3 #Printing numbers from 10 to 20 except 15 can be done using continue statement and for loop. The following code is an example of the above scenario:
4
5 for iterator in range(10, 21):
6     if iterator == 15:
7         continue
8     print(iterator)
9
10 #Python Continue Statements in while Loop
11
12 string = "JavaTpoint"
13 iterator = 0
14
15 while iterator < len(string):
16     if string[iterator] == 'a':
17         continue
18     print(string[iterator])
19     iterator += 1
20
21 #Python Continue statement in list comprehension
22
23 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
24 sq_num = [num ** 2 for num in numbers if num % 2 == 0]
25 print(sq_num)
26
```

LISTS AND TUPLES:



```
1 def example1():
2     # Using single quotes
3     str1 = "Hello Python"
4     print(str1)
5     # Using double quotes
6     str2 = "Hello Python"
7     print(str2)
8     # Using triple quotes
9     str3 = """Triple quotes are generally used for
10    represent the multiline or
11    docstring"""
12    print(str3)
13
14 def example2():
15     str = "HELLO"
16     print(str[0])
17     print(str[1])
18     print(str[2])
19     print(str[3])
20     print(str[4])
21     # It returns the IndexError because 6th index doesn't exist
22     try:
23         print(str[6])
24     except IndexError:
25         print("Index Error: string index out of range")
26
27 def example3():
28     str = "JAVATPOINT"
29     # Start 0th index to end
30     print(str[0:])
31     # Starts 1st index to 4th index
32     print(str[1:5])
33     # Starts 2nd index to 3rd index
34     print(str[2:4])
35     # Starts 4th to 2nd index
36     print(str[-3])
37     # Starts 4th to 6th index
38     print(str[4:7])
39
40 def example4():
41     str = "JAVATPOINT"
42     print(str[-1])
43     print(str[-3])
44     print(str[-2:])
45     print(str[-4:-1])
46     print(str[-7:-2])
47     # Reversing the given string
48     print(str[::-1])
```

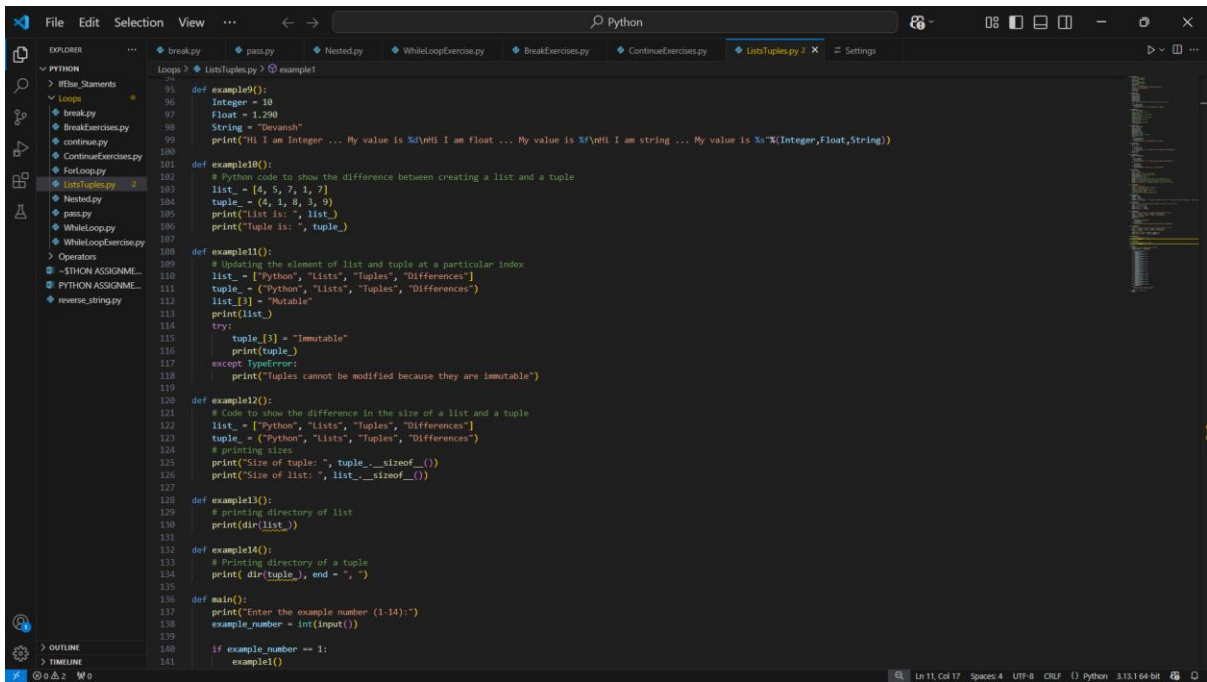
```
def example4():
    print(str[1:-1])
    try:
        print(str[-12])
    except IndexError:
        print("IndexError: string index out of range")

def example5():
    str = "HELLO"
    try:
        str[0] = "h"
    except TypeError:
        print("TypeError: 'str' object does not support item assignment")
    str = "hello"
    print(str)

def example6():
    str1 = "JAVATPOINT"
    try:
        del str1[1]
    except TypeError:
        print("TypeError: 'str' object doesn't support item deletion")
    try:
        del str1
    except NameError:
        print("NameError: name 'str1' is not defined")

def example7():
    str = "Hello"
    str1 = " world"
    print(str*3) # prints HelloHelloHello
    print(str+str1) # prints Hello world
    print(str[4]) # prints e
    print(str[2:4]) # prints ll
    print('w' in str) # prints False as w is not present in str
    print('wo' not in str1) # prints False as wo is present in str1
    print(r"C://python37") # prints C://python37 as it is written
    print("The string str : %s"%(str)) # prints The string str : Hello

def example8():
    # using triple quotes
    print("""They said, "What's there?"""")
    # escaping single quotes
    print("They said, \"What's going on?\"")
    # escaping double quotes
    print("They said, \\What's going on\\")
```



```
def example9():
    Integer = 10
    Float = 1.250
    String = "Growth"
    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))

def example10():
    # Python code to show the difference between creating a list and a tuple
    list_ = [4, 5, 7, 1, 7]
    tuple_ = (4, 1, 8, 3, 9)
    print("List is: ", list_)
    print("tuple is: ", tuple_)

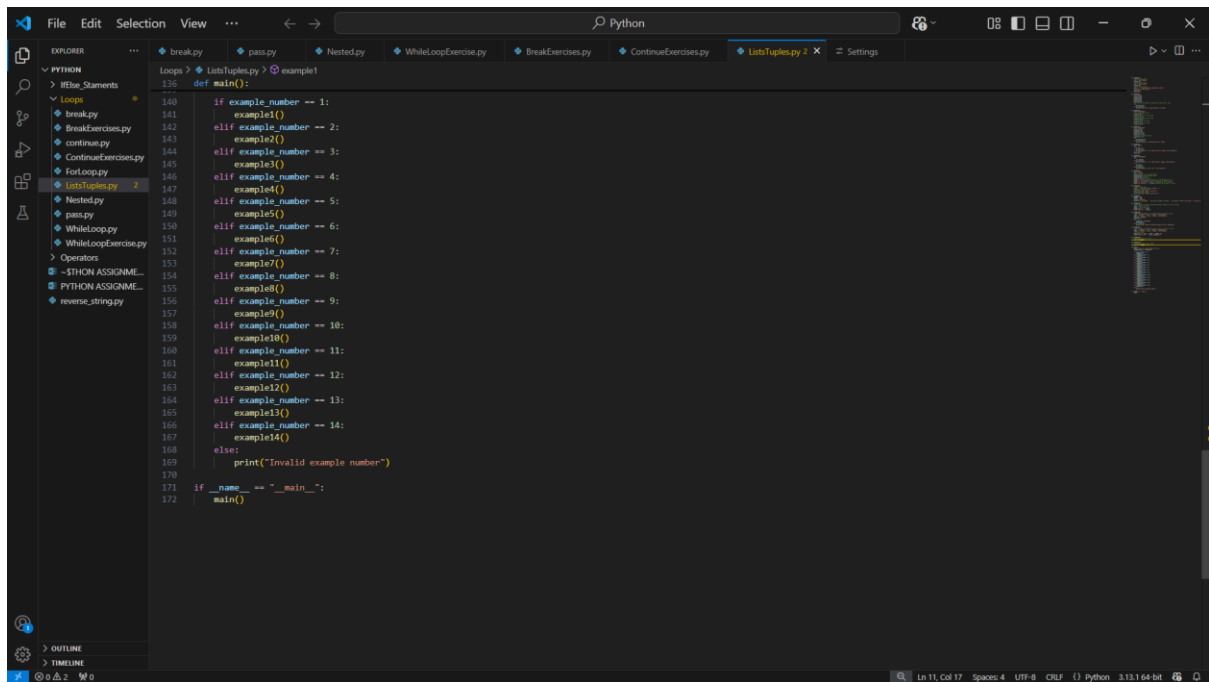
def example11():
    # Updating the element of list and tuple at a particular index
    list_ = ["Python", "Lists", "Tuples", "Differences"]
    tuple_ = ("Python", "Lists", "Tuples", "Differences")
    list_[3] = "Mutable"
    print(list_)
    try:
        tuple_[3] = "Immutable"
        print(tuple_)
    except TypeError:
        print("Tuples cannot be modified because they are immutable")

def example12():
    # Code to show the difference in the size of a list and a tuple
    list_ = ["Python", "Lists", "Tuples", "Differences"]
    tuple_ = ("Python", "Lists", "Tuples", "Differences")
    # printing size
    print("Size of tuple: ", tuple.__sizeof__())
    print("Size of list: ", list.__sizeof__())

def example13():
    # printing directory of list
    print(dir(list_))

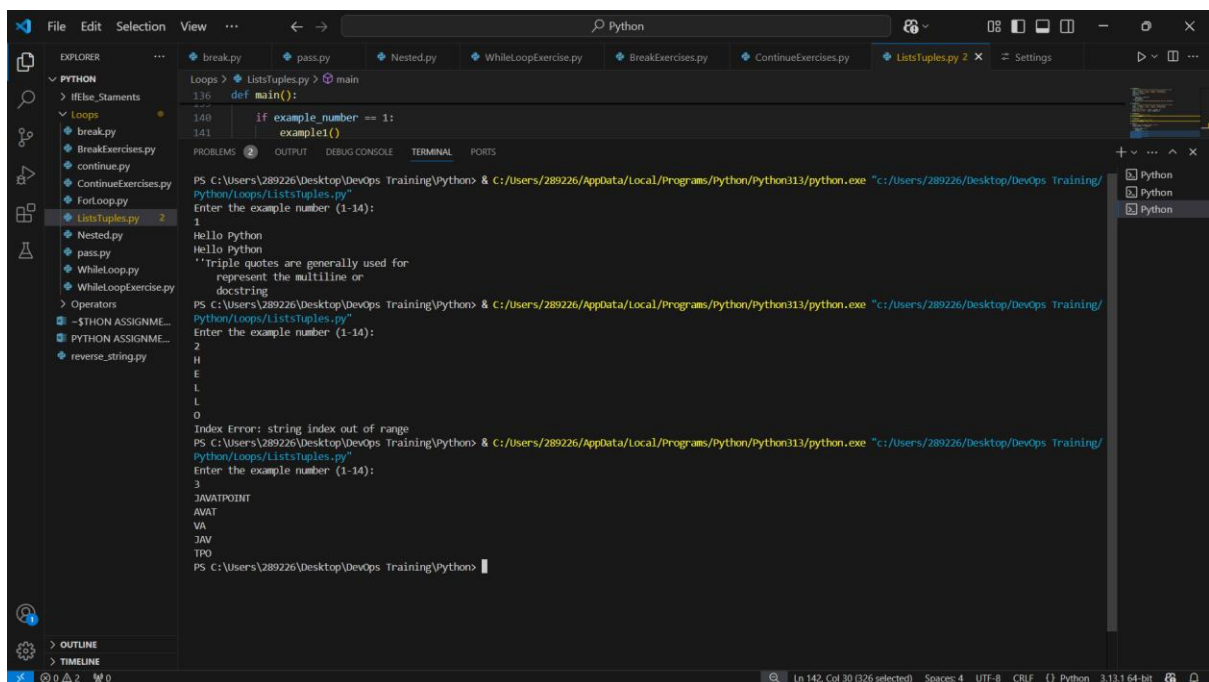
def example14():
    # printing directory of a tuple
    print(dir(tuple_))

def main():
    print("Enter the example number (1-14):")
    example_number = int(input())
    if example_number == 1:
        example1()
```



```
136 def main():
137     if example_number == 1:
138         example1()
139     elif example_number == 2:
140         example2()
141     elif example_number == 3:
142         example3()
143     elif example_number == 4:
144         example4()
145     elif example_number == 5:
146         example5()
147     elif example_number == 6:
148         example6()
149     elif example_number == 7:
150         example7()
151     elif example_number == 8:
152         example8()
153     elif example_number == 9:
154         example9()
155     elif example_number == 10:
156         example10()
157     elif example_number == 11:
158         example11()
159     elif example_number == 12:
160         example12()
161     elif example_number == 13:
162         example13()
163     elif example_number == 14:
164         example14()
165     else:
166         print("Invalid example number")
167
168 if __name__ == "__main__":
169     main()
```

OUTPUT:



```
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\289226\Desktop\DevOps Training\Python\Loops\ListsTuples.py"
Enter the example number (1-14):
1
Hello Python
Hello Python
'''triple quotes are generally used for
represent the multiline or
docstring
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\289226\Desktop\DevOps Training\Python\Loops\ListsTuples.py"
Enter the example number (1-14):
2
H
E
L
L
O
IndexError: string index out of range
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:\Users\289226\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\289226\Desktop\DevOps Training\Python\Loops\ListsTuples.py"
Enter the example number (1-14):
3
JAVATPOINT
AVAT
VA
JAV
TPO
PS C:\Users\289226\Desktop\DevOps Training\Python>
```

The screenshot displays a Windows IDE (Visual Studio Code) with a dark theme. The interface is divided into several panes:

- Explorer (Left):** Shows the file structure. The file `Python/Loops/Liststuples.py` is selected and highlighted in blue.
- Terminal (Bottom):** Displays the execution output of the Python script. The output shows the program running successfully, printing the results of a loop and a string operation.
- Code Editor (Center):** Shows the source code of `Liststuples.py`. The code defines a list `l1` and a string `str1`, then iterates over the list and prints the elements and the string.

The terminal output is as follows:

```
python.exe "C:/Users/289226/Desktop/DevOps Training/Python/Loops/Liststuples.py"
Enter the example number (1-14):
3
JAVATPOINT
AVAT
VA
JAV
TPO
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/Appdata/Local/Programs/Python/Python313/python.exe "C:/Users/289226/Desktop/DevOps Training/Python/Loops/Liststuples.py"
Enter the example number (1-14):
4
T
I
NT
OIN
ATPOI
TNIOPATAJ
IndexError: string index out of range
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/Appdata/Local/Programs/Python/Python313/python.exe "C:/Users/289226/Desktop/DevOps Training/Python/Loops/Liststuples.py"
Enter the example number (1-14):
5
TypeError: 'str' object does not support item assignment
hello
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/Appdata/Local/Programs/Python/Python313/python.exe "C:/Users/289226/Desktop/DevOps Training/Python/Loops/Liststuples.py"
Enter the example number (1-14):
7
HelloHelloHello
Hello world
o
11
False
false
C:/python37
The string str : Hello
PS C:\Users\289226\Desktop\DevOps Training\Python> & C:/Users/289226/Appdata/Local/Programs/Python/Python313/python.exe "C:/Users/289226/Desktop/DevOps Training/Python/Loops/Liststuples.py"
Enter the example number (1-14):
8
"they said, 'what's there?"
They said, "what's going on?"
```

The screenshot displays the Visual Studio Code editor with a Python file named 'reverse_string.py' open. The file contains a script that prompts the user for an example number (1-14), then asks for a string to reverse, and finally asks for a list of numbers to reverse. The terminal output shows the script's execution, including the reversed string and the reversed list. The Explorer pane on the left shows the file structure, and the Terminal pane on the right shows the command prompt and the script's output.