ASSIGNMENT-7.5

HT.NO:2303A510H6

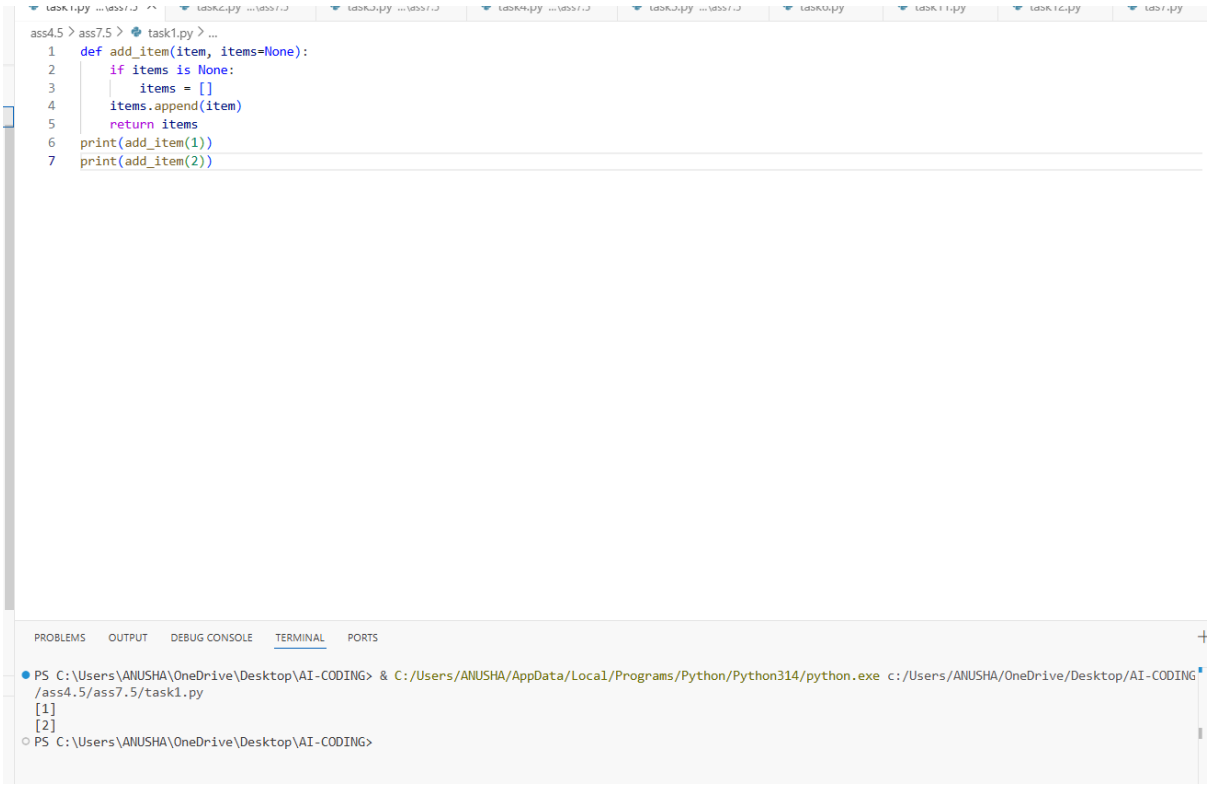
Batch No:30

Task 1: Mutable Default Argument – Function Bug

Prompt:

Analyze the given Python function where a mutable default argument causes unexpected behavior and generate a corrected version of the function.

Code & Output:



Observation:

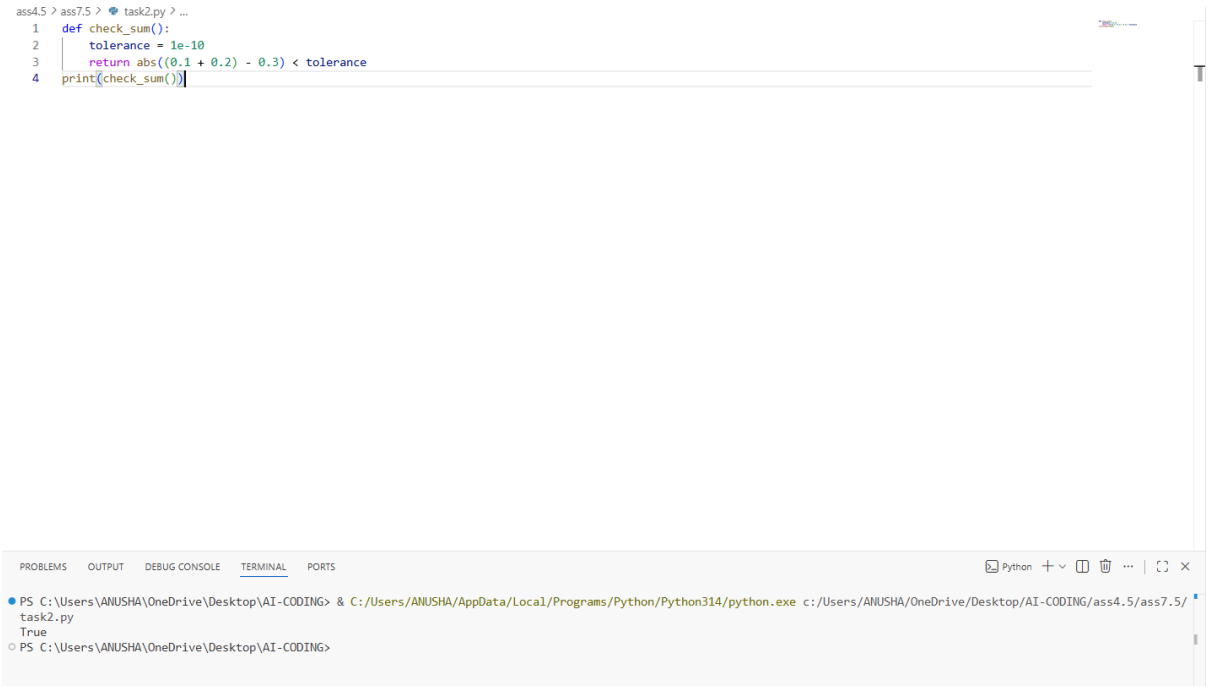
The function correctly avoids the mutable default argument bug by creating a new list for each call, resulting in independent outputs for each item added.

Task 2: Floating-Point Precision Error

Prompt:

Analyze the given Python code where a floating-point comparison fails and generate a corrected version using tolerance.

Code & Output:



Observation:

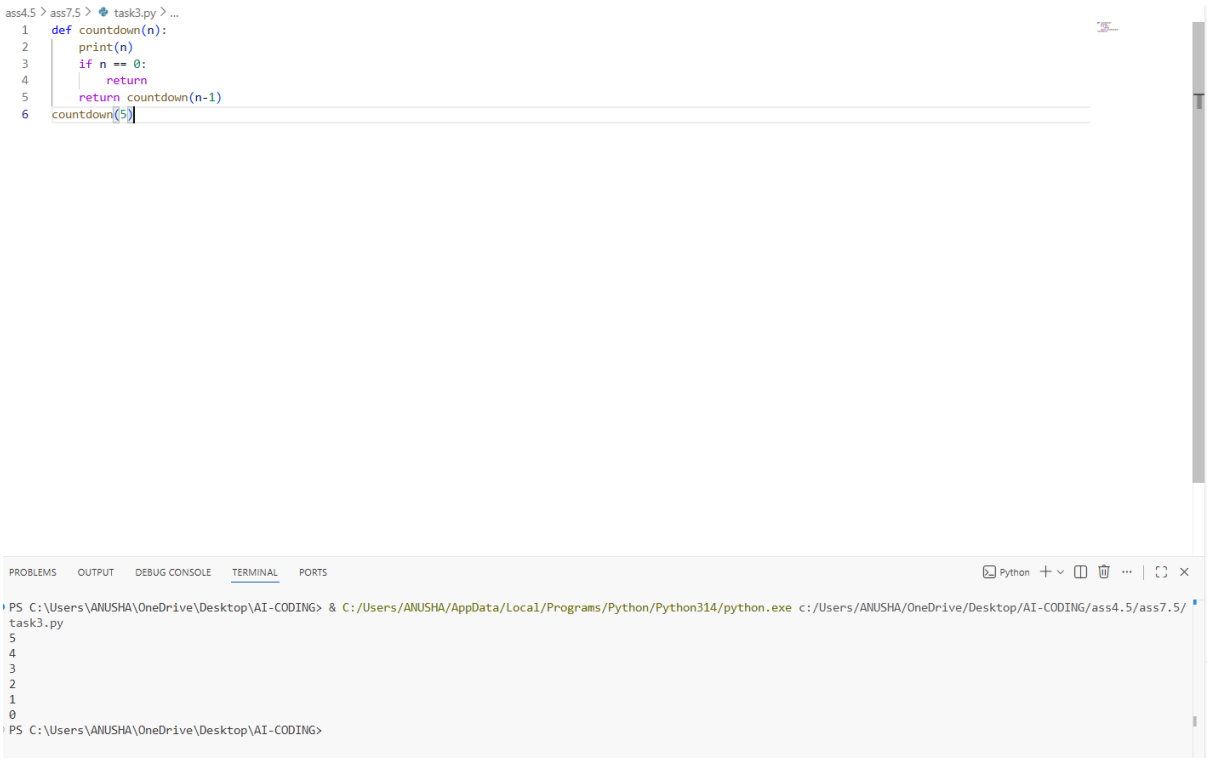
The function correctly handles floating-point precision issues by comparing values within a small tolerance instead of using direct equality.

Task 3: Recursion Error – Missing Base Case

Prompt:

Analyze the given recursive Python function that runs infinitely due to a missing base case and generate a corrected version with a proper stopping condition.

Code & Output:



Observation:

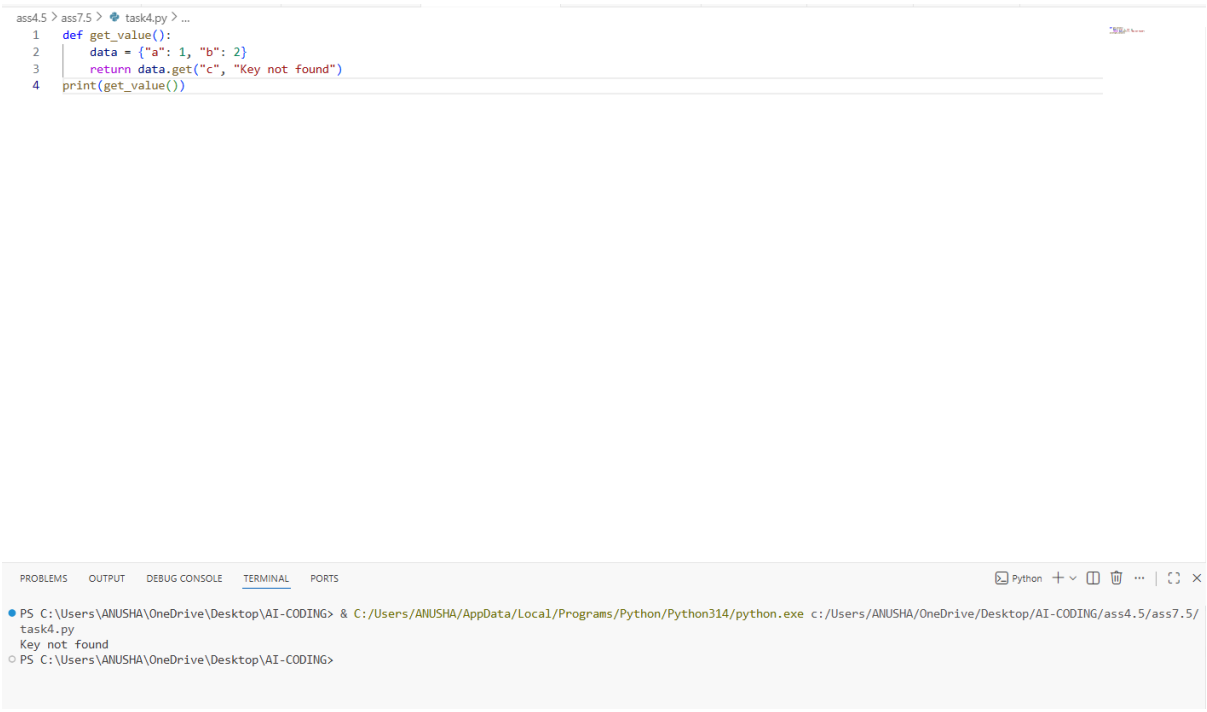
The recursion now stops correctly due to the added base case.
Each function call reduces the value of n, preventing infinite recursion.
The program executes safely and produces the expected countdown output.

Task 4: Dictionary Key Error

Prompt:

Fix the Key Error caused by accessing a missing dictionary key using .get() or error handling.

Code& Output:



Observation:

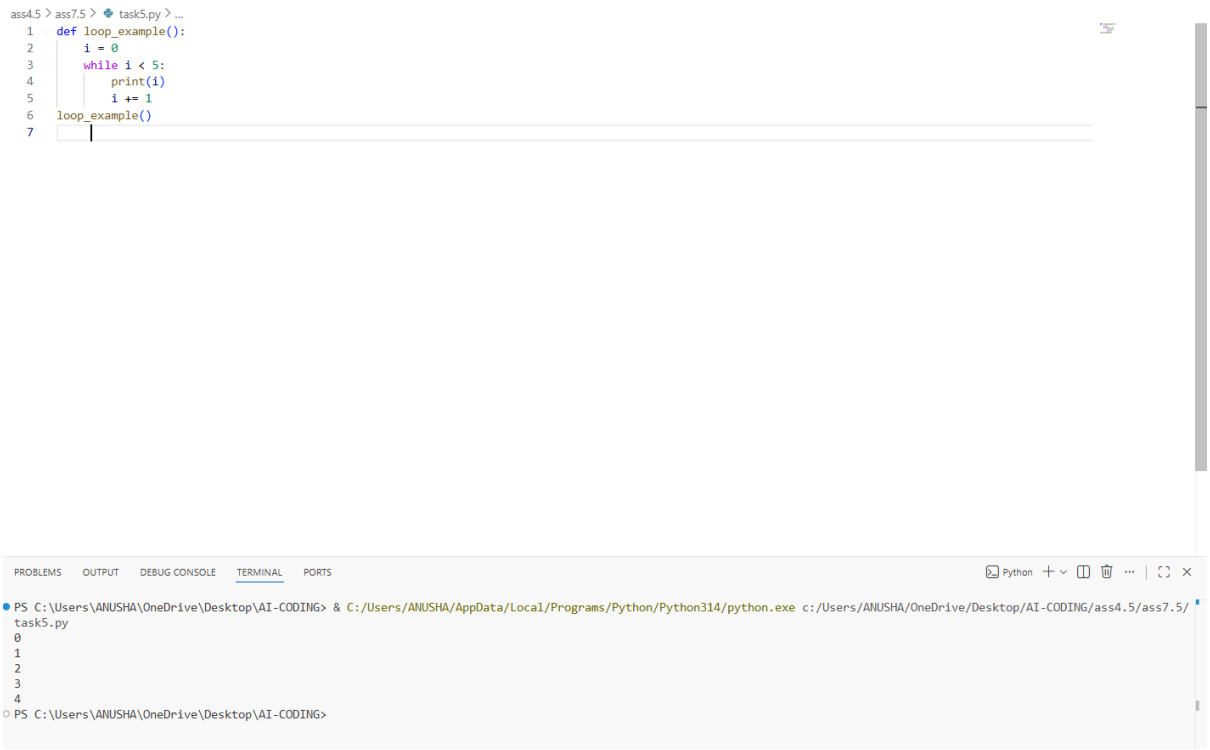
Since the key "c" is not available in the dictionary, the program prints "Key not found" instead of a number.

Task 5: Infinite Loop – Wrong Condition

Prompt:

Fix the infinite loop by updating the loop variable so the condition eventually becomes false.

Code& Output:



Observation:

The loop starts from 0 and prints the numbers 0, 1, 2, 3, and 4 because it runs while the value of i is less than 5. The value of i increases by 1 each time, so the loop stops automatically when i reaches 5.

Task 6: Unpacking Error – Wrong Variables

Prompt:

Fix the tuple unpacking error by matching the number of variables to the values or ignoring extra values using _ .

Code& Output:

```
ass4.5 > ass7.5 > task6.py > ...
1  a, b, c = (1, 2, 3)
2  print(a)
3  print(b)
4  print(c)
5
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   Python

PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING> & C:/Users/ANUSHA/AppData/Local/Programs/Python/Python314/python.exe c:/Users/ANUSHA/OneDrive/Desktop/AI-CODING/ass4.5/ass7.5/
task6.py
1
2
3
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING>
```

Observation:

The values 1, 2, and 3 are assigned to variables a, b, and c in a single statement. The program prints 1, 2, and 3 on separate lines.

Task 7: Mixed Indentation – Tabs vs Spaces

Prompt:

Fix the Python code by correcting mixed indentation (tabs vs spaces) so it executes without errors.

Code& Output:

```
ass4.5 > ass7.5 > tas7.py > ...
1  # Fix the Python code by correcting mixed indentation (tabs vs spaces) so it executes without errors
2  def func():
3      x = 5
4      y = 10
5      return x + y
6  result = func()
7  print(result)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Python +

PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING> & C:/Users/ANUSHA/AppData/Local/Programs/Python/Python314/python.exe c:/Users/ANUSHA/OneDrive/Desktop/AI-C
tas7.py
15
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING>
```

Observation:

The program failed due to inconsistent indentation inside the function block. After applying consistent indentation using spaces, the code executed successfully and produced the correct output.

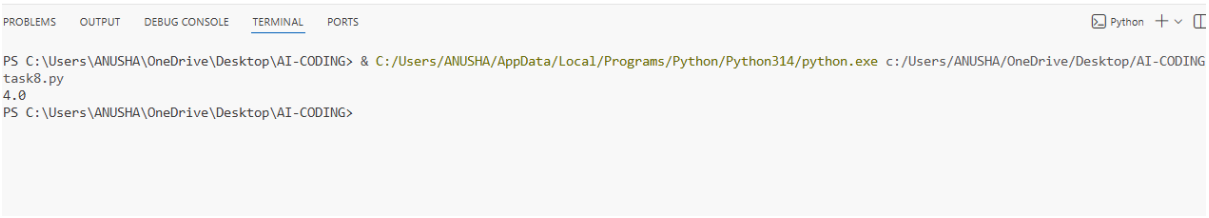
Task 8: Import Error – Wrong Module Usage

Prompt:

Fix the Python code by correcting the wrong module import so it runs without errors.

Code&Output:

```python
#Fix the Python code by correcting the wrong module import so it runs without errors
import math
print(math.sqrt(16))
```



Observation:

The program produced an import error because an incorrect module name was used. After replacing it with the correct built-in “math” module, the code executed successfully and produced the expected output.

Task 9: Unreachable Code – Return Inside Loop

Prompt:

Fix the Python code by moving the return statement outside the loop so the function correctly calculates the total sum.

Code& Output:

```python
#Fix the Python code by moving the return statement outside the loop so the function correctly calculates the total s
def total(numbers):
    total_sum = 0
    for n in numbers:
        total_sum += n
    return total_sum
print(total([1, 2, 3]))
```

Observation:

The program returned a value during the first loop iteration due to a return statement inside the loop. After moving the return statement outside the loop and accumulating values properly, the function iterated through all elements and produced the correct total.

Task 10: Name Error – Undefined Variable

Prompt:

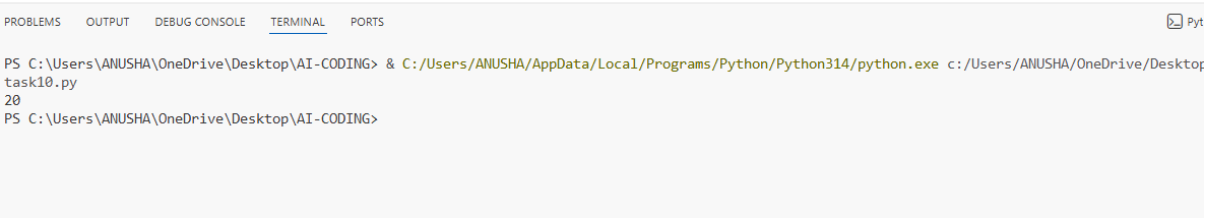
Analyze the given Python code where a variable is used before being defined, identify the error, and fix it by correcting the variable definition.

Code&Output:

```python

def calculate_area(length, width):
    return length * width
print(calculate_area(5, 4))


```



Observation:

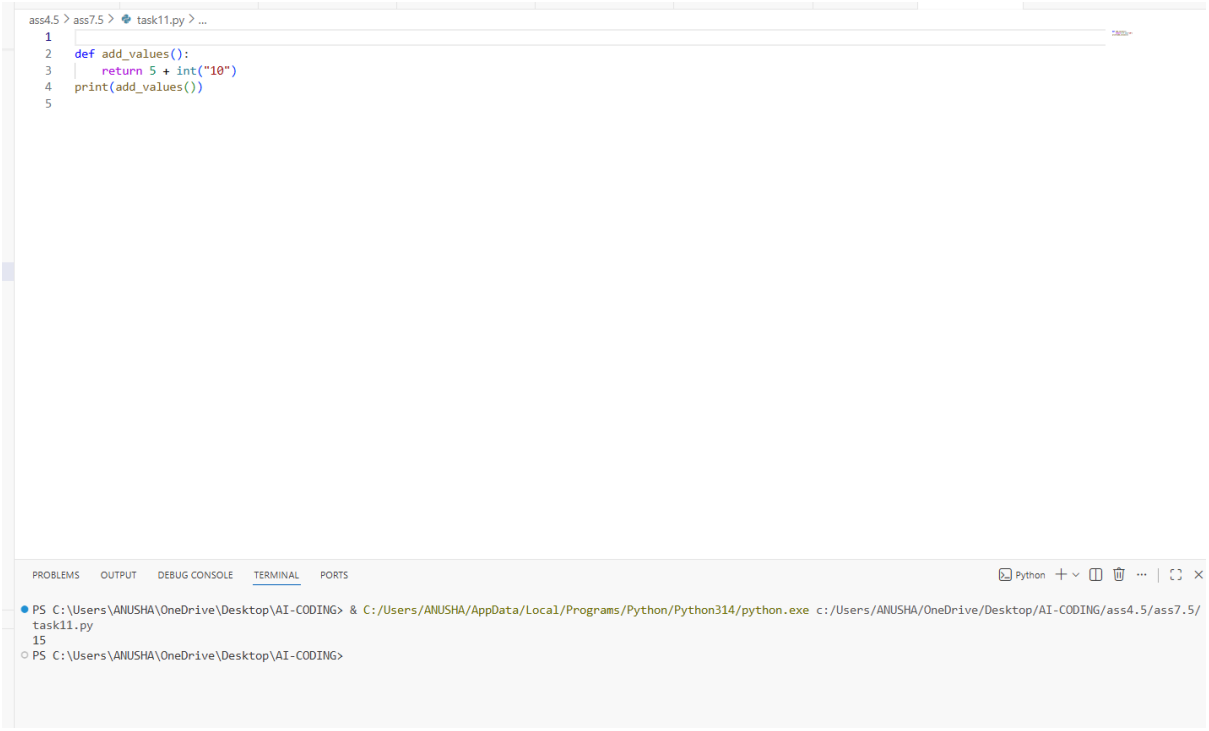
The original code produced a NameError due to undefined variables. After passing length and width as parameters, the error was resolved and all assert test cases executed successfully.

Task 11: Type Error – Mixing Data Types Incorrectly

Prompt:

Analyze the given Python code that performs addition using type conversion, explain how the operation works, and verify that the program executes without errors.

Code& Output:



Observation:

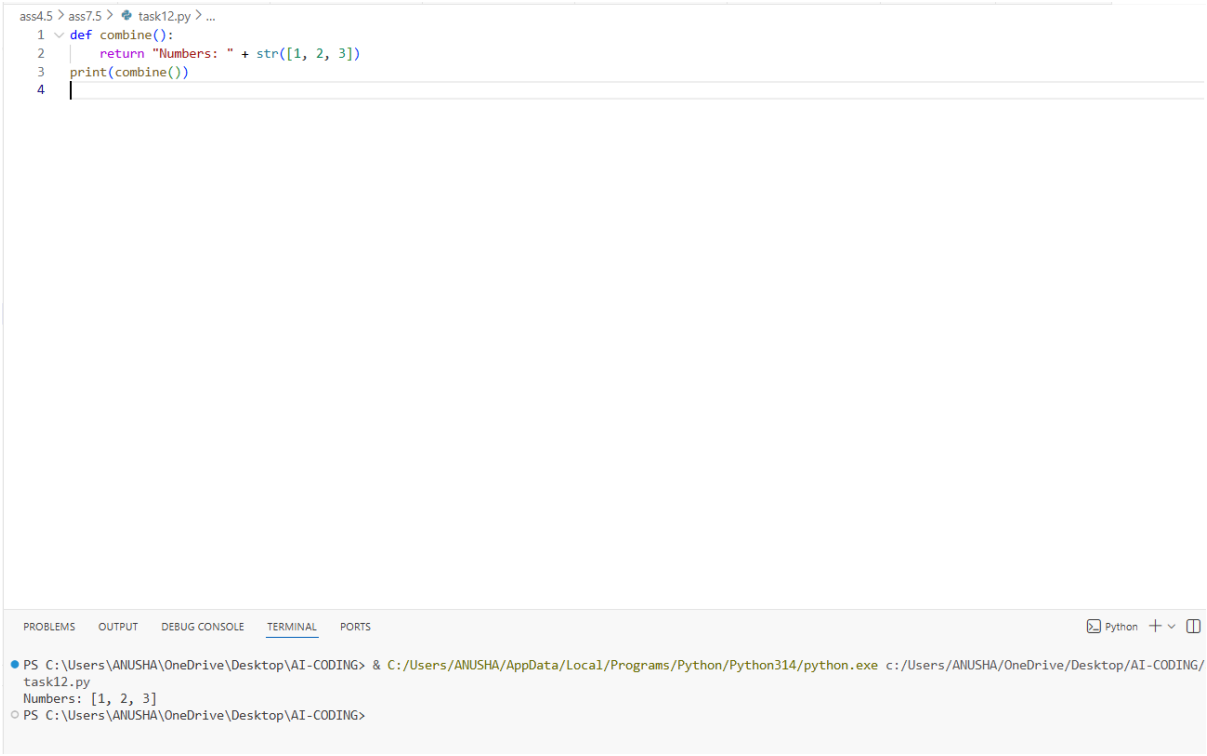
The code executes successfully without any error. The string value "10" is converted into an integer using int(), allowing valid addition with the integer 5, and the final output is 15.

Task 12: Type Error – String + List Concatenation

Prompt:

Analyze the Python code that converts a list to a string for concatenation, explain the behavior, and observe the output.

Code&Output:



Observation:

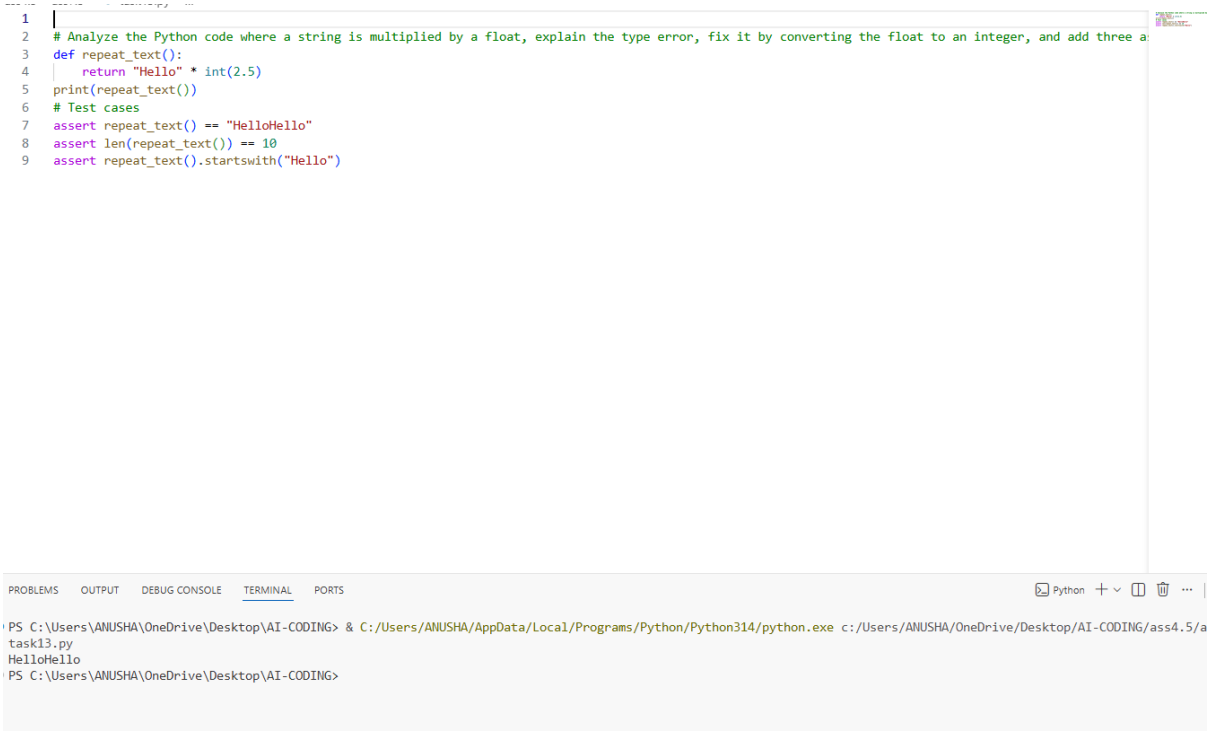
The code executes successfully without any errors. The list [1, 2, 3] is converted into a string using str(), allowing it to be concatenated with the string "Numbers: ". The output displayed is Numbers: [1, 2, 3].

Task 13: Type Error – Multiplying String by Float

Prompt:

Analyze the Python code where a string is multiplied by a float, explain the type error, fix it by converting the float to an integer, and add three assert-based test cases.

Code& Output:



```
# Analyze the Python code where a string is multiplied by a float, explain the type error, fix it by converting the float to an integer, and add three a
def repeat_text():
    return "Hello" * int(2.5)
print(repeat_text())
# Test cases
assert repeat_text() == "HelloHello"
assert len(repeat_text()) == 10
assert repeat_text().startswith("Hello")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Python

PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING> & C:/Users/ANUSHA/AppData/Local/Programs/Python/Python314/python.exe c:/Users/ANUSHA/OneDrive/Desktop/AI-CODING/ass4.5/a
task13.py
HelloHello
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING>
```

Observation:

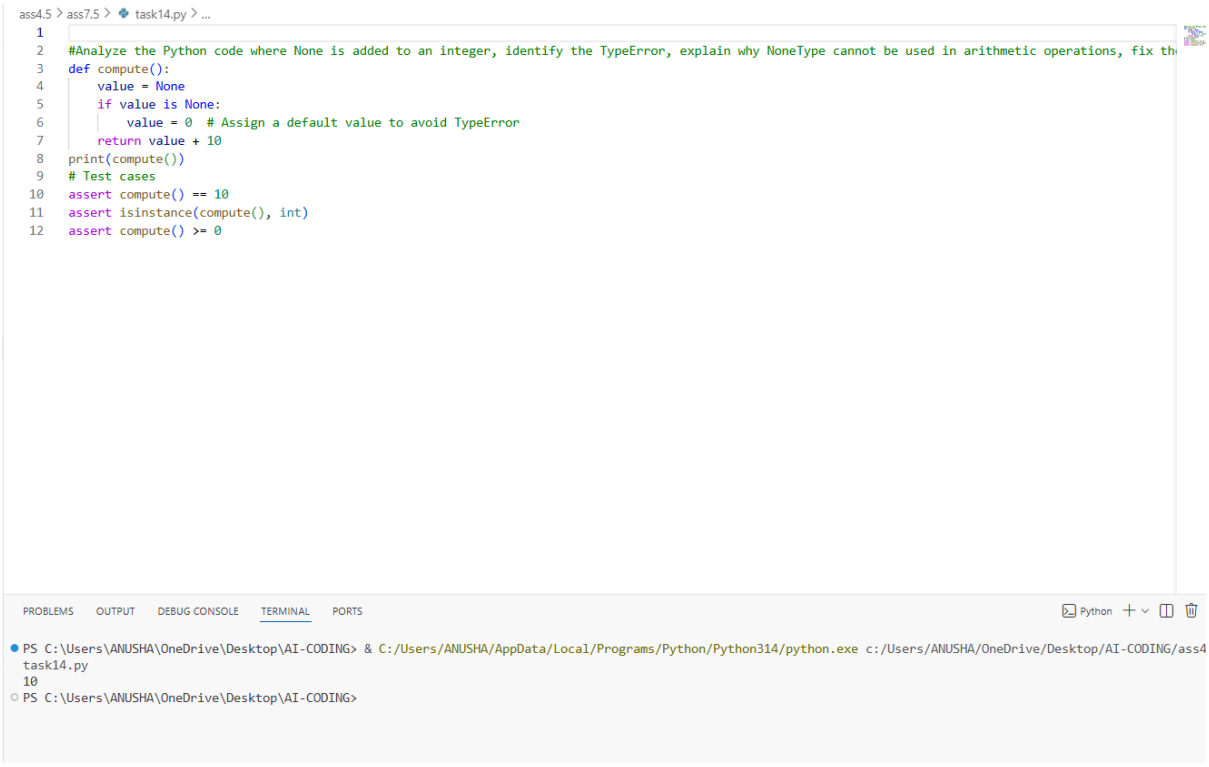
The program raised a TypeError because Python does not allow multiplication of a string by a float. Strings can only be multiplied by integers to indicate repetition. After converting the float value to an integer, the code executed successfully and produced the expected output.

Task 14: Type Error – Adding None to Integer

Prompt:

Analyze the Python code where None is added to an integer, identify the TypeError, explain why NoneType cannot be used in arithmetic operations, fix the issue by assigning a default value, and validate the fix using assert statements.

Code& Output:



Observation:

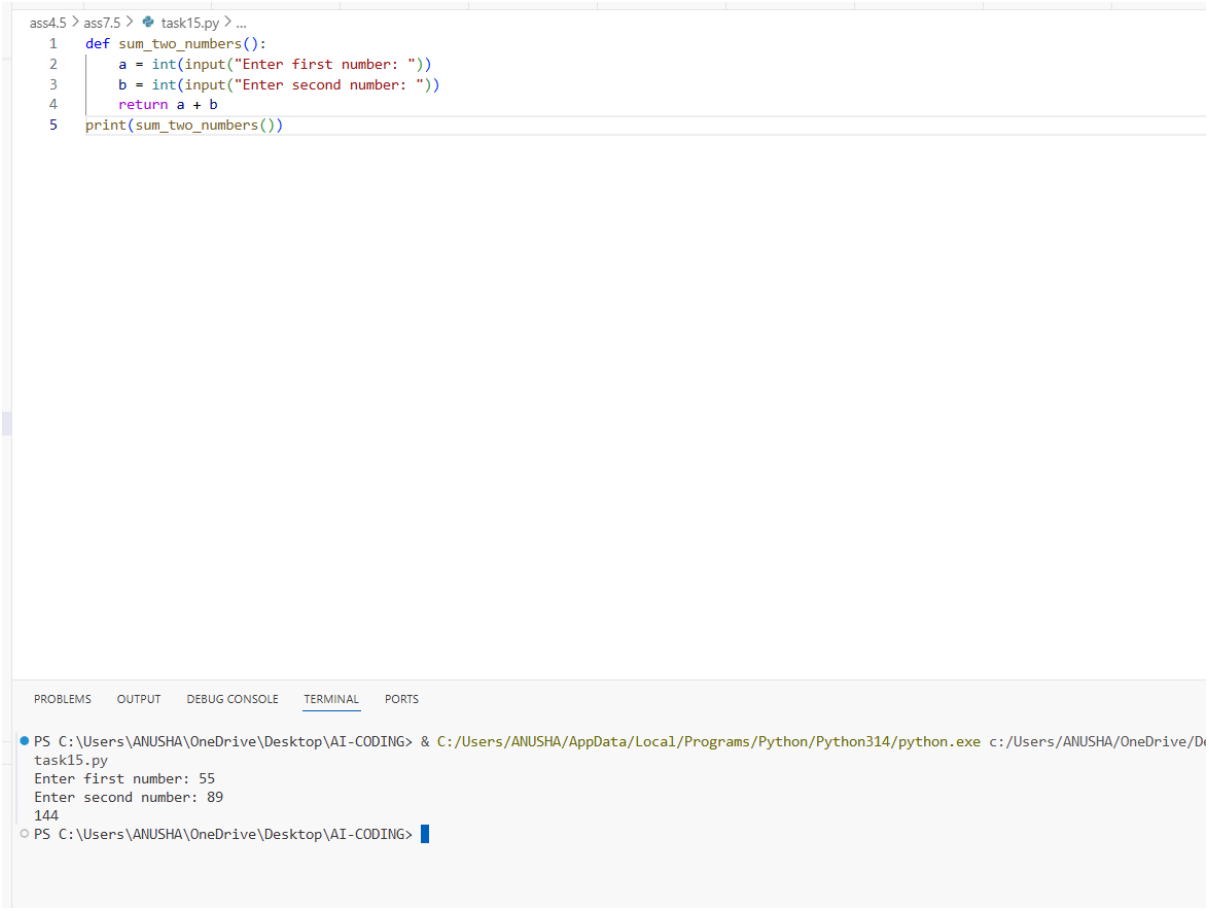
The program raised a TypeError because NoneType does not support arithmetic operations with integers. Since None represents the absence of a value, it cannot be added to a number. By assigning a valid default integer value, the computation executed successfully and produced the correct output.

Task 15: Type Error – Input Treated as String Instead of Number

Prompt:

Fix the code by converting user input from string to integer using int() so the numbers are added correctly instead of concatenated.

Code&Output:



Observation:

The program asks the user to enter two numbers and changes them into whole numbers. Then it adds those two numbers together and shows the total.