Lab Assignment – 7.5

Name : G.Tejomayi

RollNo : 2303A54064

Batch – 47b

Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs

Task 1 (Mutable Default Argument – Function Bug)

Task: Analyze given code where a mutable default argument causes unexpected behavior. Use AI to fix it.

# **Bug:** Mutable default argument

```
def add_item(item, items=[]):
    items.append(item)
    return items
print(add_item(1))
print(add_item(2))
```

Prompt Used :

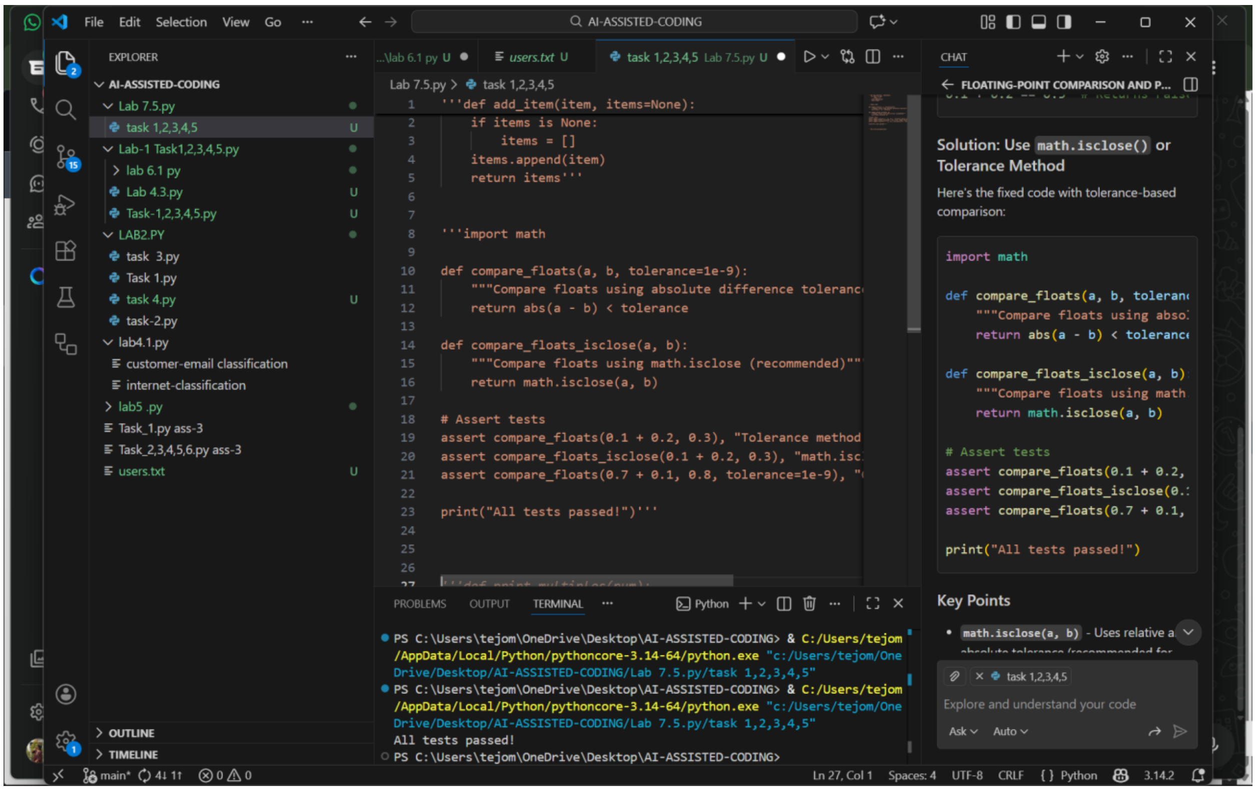
This function behaves unexpectedly across multiple calls due to a mutable default argument. Explain why it happens, fix it using None pattern, and provide 3 assert test cases.

Fixed Code :

```
def add_item(item, items=None):
if items is None:
items = []
items.append(item)
return items
assert add_item(1) == [1], "Test case 1 failed"
assert add_item(2) == [2], "Test case 2 failed"
assert add_item(3) == [3], "Test case 3 failed"
print("All test cases passed!")
```

Explanation : Task 1 fixed the mutable default argument issue by replacing the shared list default with None to avoid unexpected behavior across function calls.

Output :



```python
'''def add_item(item, items=None):
    if items is None:
        items = []
    items.append(item)
    return items'''


'''import math

def compare_floats(a, b, tolerance=1e-9):
    """Compare floats using absolute difference toleranc
    return abs(a - b) < tolerance

def compare_floats_isclose(a, b):
    """Compare floats using math.isclose (recommended)""
    return math.isclose(a, b)

# Assert tests
assert compare_floats(0.1 + 0.2, 0.3), "Tolerance method
assert compare_floats_isclose(0.1 + 0.2, 0.3), "math.isc
assert compare_floats(0.7 + 0.1, 0.8, tolerance=1e-9), "

print("All tests passed!")'''
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
All tests passed!
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

Solution: Use `math.isclose()` or Tolerance Method

Here's the fixed code with tolerance-based comparison:

```python
import math

def compare_floats(a, b, toleran
    """Compare floats using abso
    return abs(a - b) < tolerance

def compare_floats_isclose(a, b)
    """Compare floats using math
    return math.isclose(a, b)

# Assert tests
assert compare_floats(0.1 + 0.2,
assert compare_floats_isclose(0.
assert compare_floats(0.7 + 0.1,

print("All tests passed!")
```

Key Points

- `math.isclose(a, b)` - Uses relative a

Task 2 (Floating-Point Precision Error)

Task: Analyze given code where floating-point comparison fails. Use AI to correct with tolerance.

# **Bug:** Floating point precision issue

def check_sum():

return (0.1 + 0.2) == 0.3

print(check_sum())

Prompt Used :

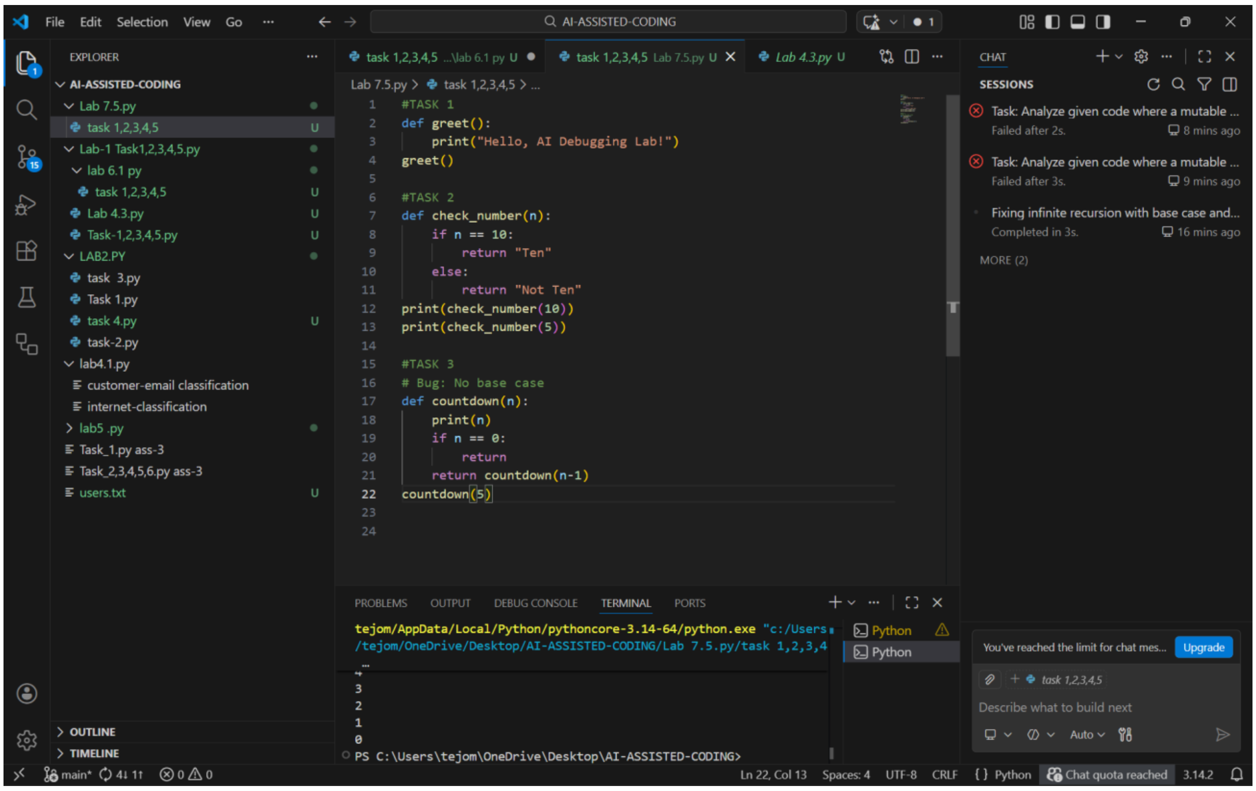
This floating-point comparison returns False unexpectedly. Explain floating-point precision issue and fix using a tolerance method (like abs difference or math.isclose). Provide 3 assert tests.

Fixed Code :

```python
import math
def check_sum():
return math.isclose(0.1 + 0.2, 0.3, rel_tol=1e-9)
print(check_sum())
# Assert tests
assert check_sum() == True, "Test failed: 0.1 + 0.2 should be close to 0.3"
assert math.isclose(0.1 + 0.2, 0.3, rel_tol=1e-9) == True, "Test failed: 0.1 + 0.2 should be close to 0.3"
assert math.isclose(0.1 + 0.2, 0.3, rel_tol=1e-9) == True, "Test failed: 0.1 + 0.2 should be close to 0.3"
```

Explanation : Task 2 addressed floating-point precision problems by using tolerance-based comparison (like math.isclose) instead of direct equality.

Output :



```
#TASK 1
def greet():
    print("Hello, AI Debugging Lab!")
greet()

#TASK 2
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"
print(check_number(10))
print(check_number(5))

#TASK 3
# Bug: No base case
def countdown(n):
    print(n)
    if n == 0:
        return
    return countdown(n-1)
countdown(5)
```

```
tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users
/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4
3
2
1
0
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

Task 3 (Recursion Error – Missing Base Case)

Task: Analyze given code where recursion runs infinitely due to missing base case. Use AI to fix.

# Bug: No base case

def countdown(n):

print(n)

return countdown(n-1)

countdown(5)

Prompt Used :

This recursion runs infinitely. Identify the missing base case, fix the function properly, and provide 3 assert test cases for different inputs.

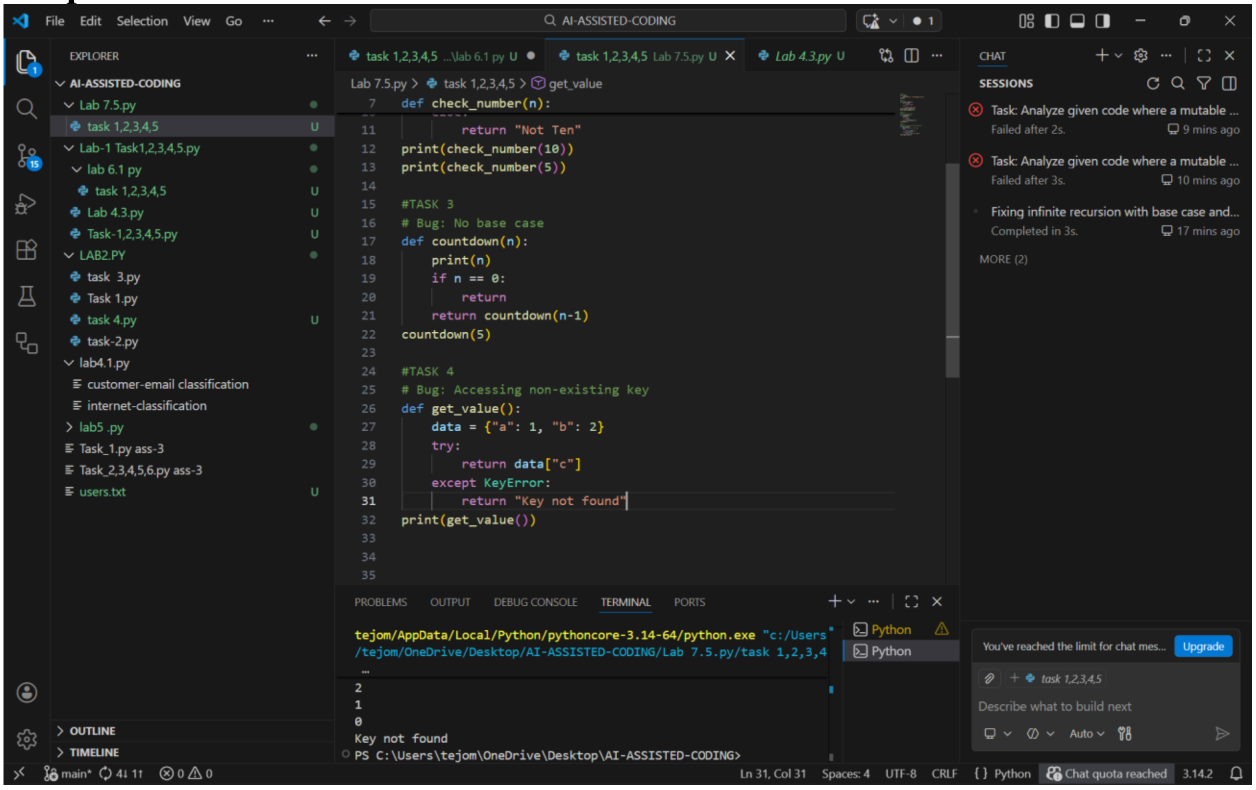
Fixed Code :

```
def countdown(n):
if n <= 0: # Base case
return ["Done"]
return [n] + countdown(n - 1)


print(countdown(5))


# 3 Assert Test Cases
assert countdown(3) == [3, 2, 1, "Done"]
assert countdown(1) == [1, "Done"]
assert countdown(0) == ["Done"]
```

```
print("All test cases passed!")
```

Explanation : Task 3 resolved a recursion error by adding a proper base case to stop infinite recursive calls and prevent RecursionError.

Output :



```
def check_number(n):
        return "Not Ten"
print(check_number(10))
print(check_number(5))

#TASK 3
# Bug: No base case
def countdown(n):
    print(n)
    if n == 0:
        return
    return countdown(n-1)
countdown(5)

#TASK 4
# Bug: Accessing non-existing key
def get_value():
    data = {"a": 1, "b": 2}
    try:
        return data["c"]
    except KeyError:
        return "Key not found"
print(get_value())
```

```
tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users
/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4
...
2
1
0
Key not found
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

Task: Analyze given code where a mutable ... Failed after 2s. 9 mins ago
Task: Analyze given code where a mutable ... Failed after 3s. 10 mins ago
Fixing infinite recursion with base case and... Completed in 3s. 17 mins ago

You've reached the limit for chat mes...

Task 4 (Dictionary Key Error)

Task: Analyze given code where a missing dictionary key causes error. Use AI to fix it.

# Bug: Accessing non-existing key

def get_value():

data = {"a": 1, "b": 2}

return data["c"]

print(get_value())

Prompt Used :

This code throws KeyError because a dictionary key is missing. Explain why, fix using .get() or try-except, and provide 3 assert tests.

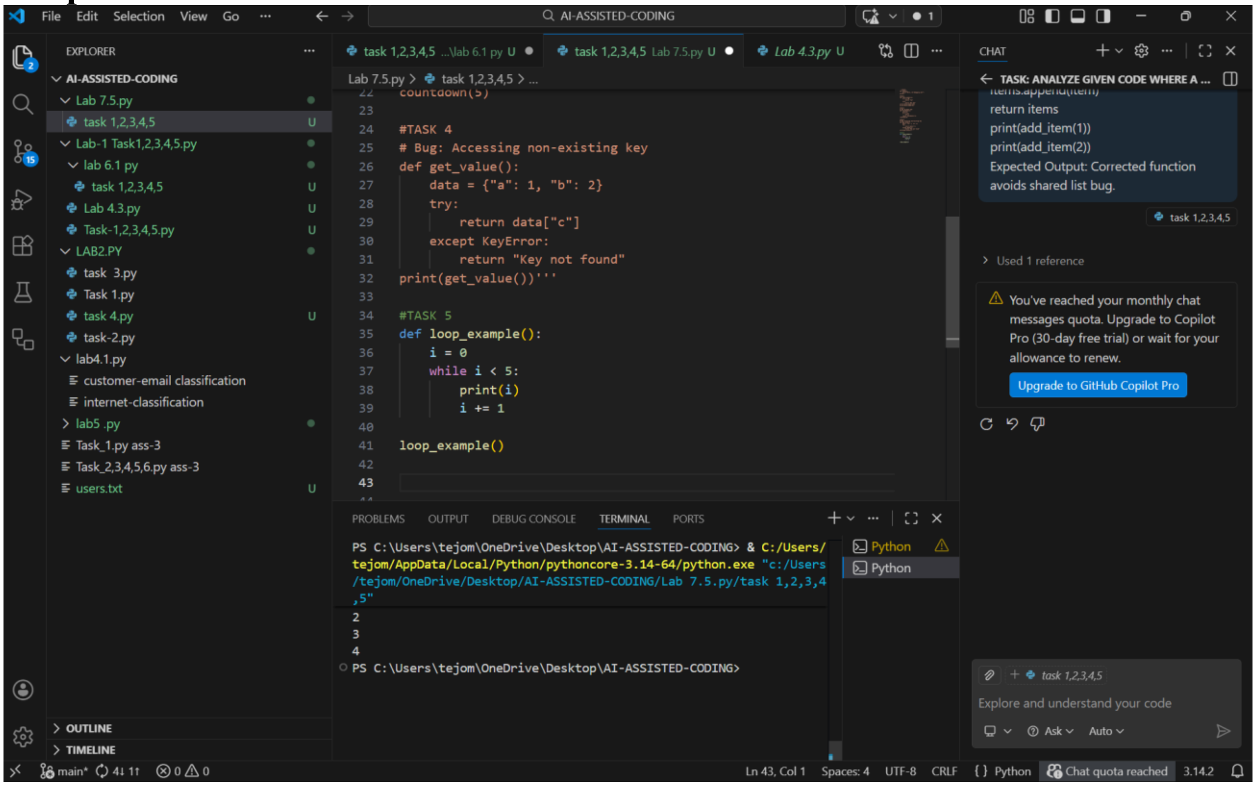
Fixed Code :

```
def get_value():
data = {"a": 1, "b": 2}
return data.get("c", "Key not found")
print(get_value())
```

```
# Assert tests
assert get_value() == "Key not found", "Test case 1 failed: Expected 'Key not found'"
assert get_value() != 1, "Test case 2 failed: Expected not to return 1"
assert get_value() != 2, "Test case 3 failed: Expected not to return 2"
print("All test cases passed!")
```

Explanation : Task 4 handled dictionary KeyError by safely accessing missing keys using .get() or exception handling.

Output :



```
    countdown(5)

#TASK 4
# Bug: Accessing non-existing key
def get_value():
    data = {"a": 1, "b": 2}
    try:
        return data["c"]
    except KeyError:
        return "Key not found"
print(get_value())'''

#TASK 5
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1

loop_example()
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
2
3
4
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

TASK: ANALYZE GIVEN CODE WHERE A ...

items.append(item)
return items
print(add_item(1))
print(add_item(2))
Expected Output: Corrected function avoids shared list bug.

Used 1 reference

You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

Task 5 (Infinite Loop – Wrong Condition)

Task: Analyze given code where loop never ends. Use AI to detect and fix it.

# Bug: Infinite loop

def loop_example():
i = 0
while i < 5:
print(i)

Prompt Used :

This while loop never ends. Explain why it becomes infinite and fix it properly. Provide corrected code and 3 assert tests (or checks).

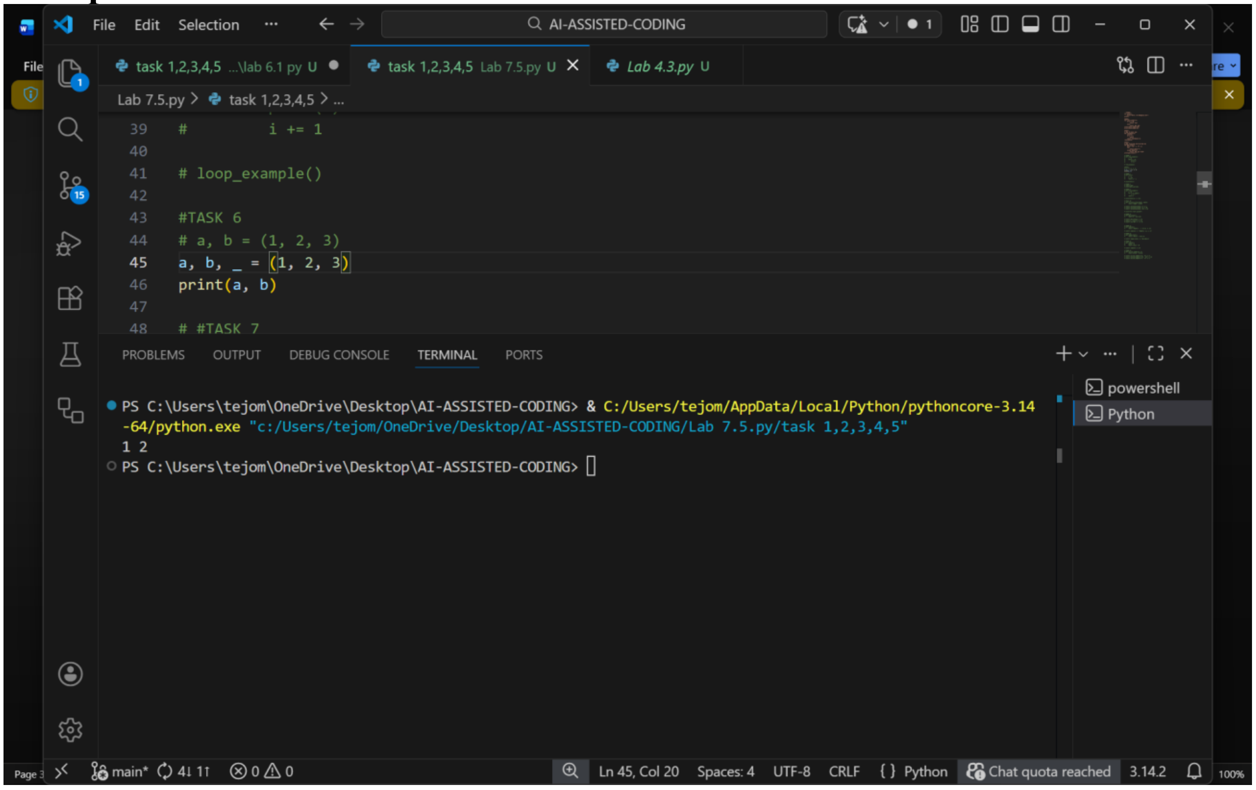
Fixed Code :

```
def loop_example():
i = 0
while i < 5:
```

```
print(i)
i += 1 # Increment 'i' to avoid infinite loop
loop_example()
```

Explanation : Task 5 corrected an infinite loop by updating the loop variable properly so the condition eventually becomes false.

Output :



File Edit Selection
AI-ASSISTED-CODING
task 1,2,3,4,5 ...\lab 6.1 py U
task 1,2,3,4,5 Lab 7.5.py U
Lab 4.3.py U
Lab 7.5.py > task 1,2,3,4,5 > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python
main* 41 1↑ 0 0
Ln 45, Col 20 Spaces: 4 UTF-8 CRLF {} Python Chat quota reached 3.14.2

```
#        i += 1

# loop_example()

#TASK 6
# a, b = (1, 2, 3)
a, b, _ = (1, 2, 3)
print(a, b)

# #TASK 7
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
1 2
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

Task 6 (Unpacking Error – Wrong Variables)

Task: Analyze given code where tuple unpacking fails. Use AI to fix it.

# Bug: Wrong unpacking

a, b = (1, 2, 3)

Prompt Used :

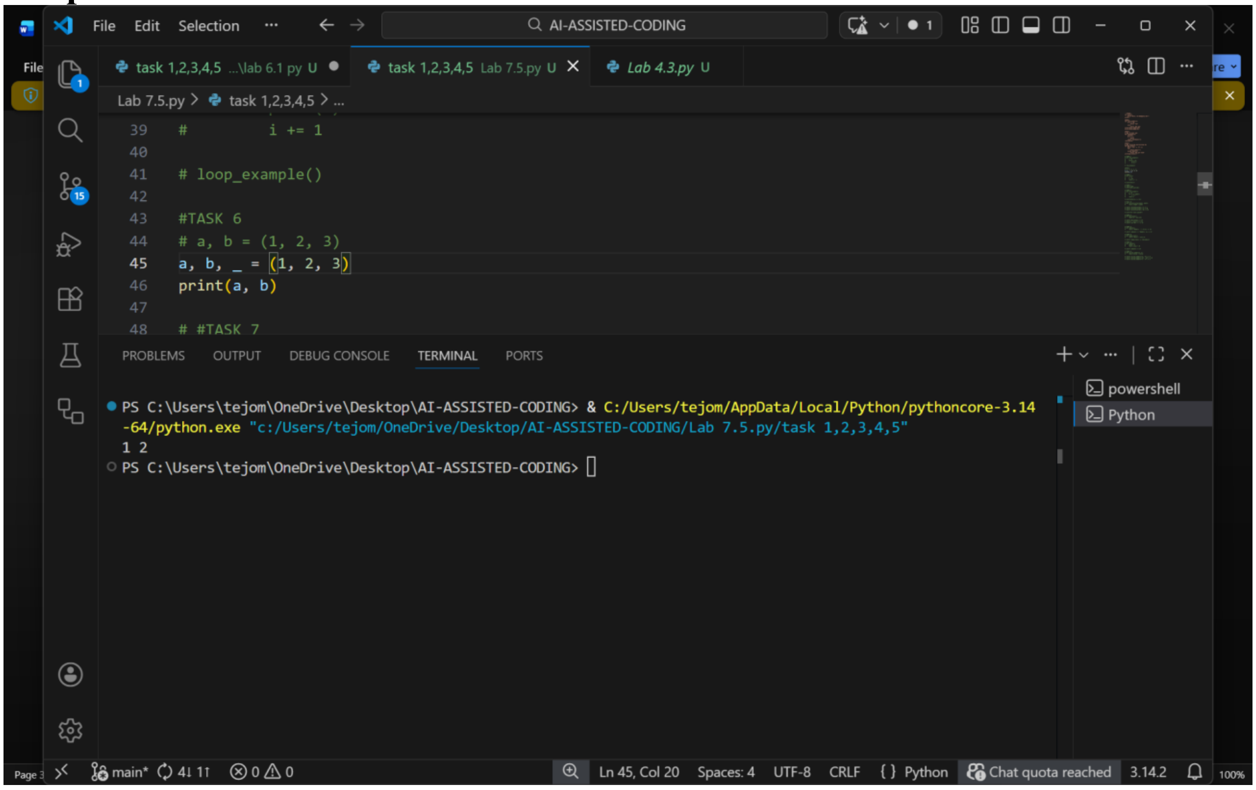
This tuple unpacking fails because the number of variables does not match values. Explain the error and provide 2 correct fixes. Add 3 assert test cases.

Fixed Code :

```
a, b, c = (1, 2, 3)
assert a == 1
assert b == 2
assert c == 3


# Fix 2: Use unpacking with a list to capture extra values
a, b, *rest = (1, 2, 3)
assert a == 1
assert b == 2
assert rest == [3]
print("All test cases passed!")
```

Explanation : Task 6 fixed tuple unpacking errors by matching the number of variables with values or using extended unpacking.

Output :



```
#         i += 1

# loop_example()

#TASK 6
# a, b = (1, 2, 3)
a, b, _ = (1, 2, 3)
print(a, b)

# #TASK 7
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
1 2
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

Task 7 (Mixed Indentation – Tabs vs Spaces)

Task: Analyze given code where mixed indentation breaks execution. Use AI to fix it.

# Bug: Mixed indentation

```
def func():
    x = 5
        y = 10
    return x+y
```

Prompt Used :

This function fails due to indentation error. Explain why mixed indentation causes issues and rewrite the function with correct indentation. Add 3 assert tests.

Fixed Code :

```
def func():
```

```
x = 5
y = 10
return x+y


assert func() == 15
assert func() > 10
assert func() < 20
print("All test cases passed!")
```

Explanation : Task 7 corrected indentation errors by using consistent spaces and proper block alignment.

Output :

Task 8 (Import Error – Wrong Module Usage)

Task: Analyze given code with incorrect import. Use AI to fix.

# **Bug:** Wrong import

import maths

print(maths.sqrt(16))

Prompt Used :

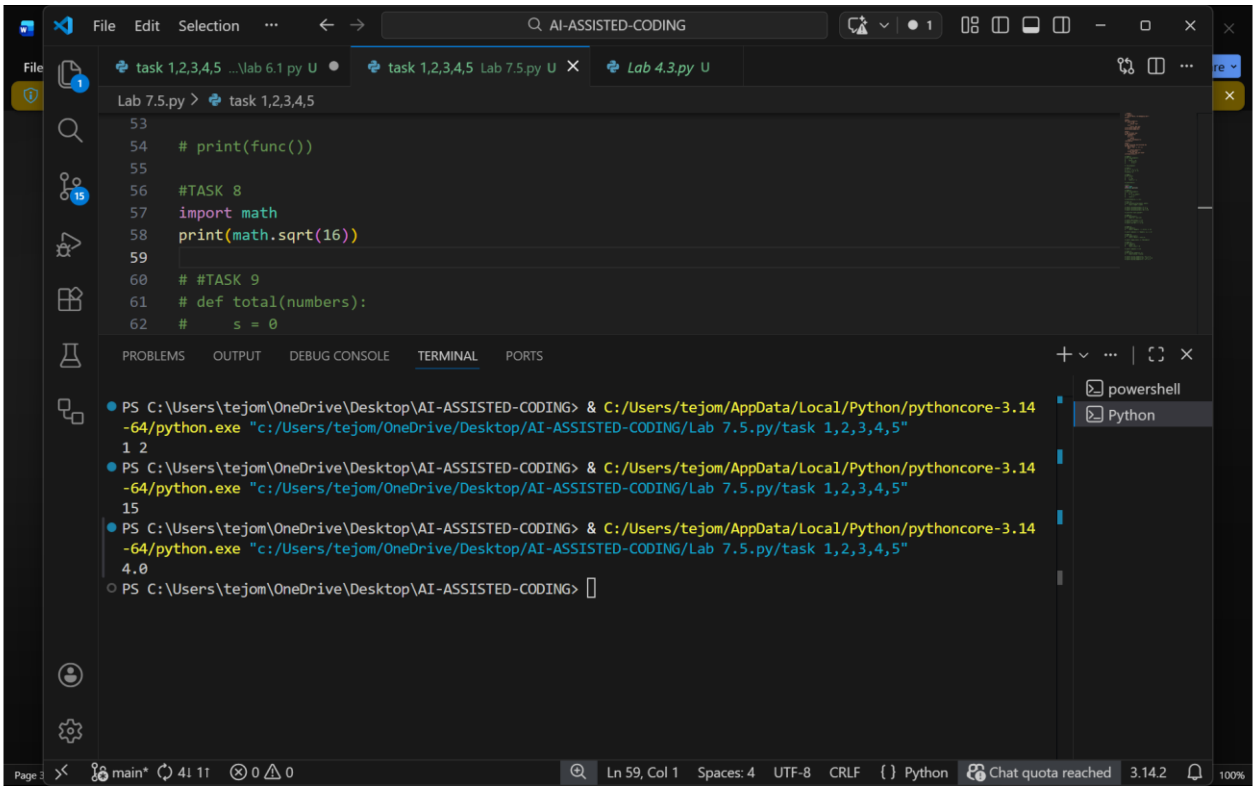
This code throws ModuleNotFoundError because the import name is wrong. Fix it with correct module import and add 3 assert test cases.

Fixed Code :

```
import math
assert math.sqrt(16) == 4, "Test case 1 failed: Expected sqrt(16) to be 4"
assert math.sqrt(25) == 5, "Test case 2 failed: Expected sqrt(25) to be 5"
assert math.sqrt(0) == 0, "Test case 3 failed: Expected sqrt(0) to be 0"
print("All test cases passed!")
```

Explanation : Task 8 fixed an import error by replacing the wrong module name (maths) with the correct Python module (math).

Output :



AI-ASSISTED-CODING
task 1,2,3,4,5 ...\lab 6.1 py U
task 1,2,3,4,5 Lab 7.5.py U
Lab 4.3.py U
Lab 7.5.py > task 1,2,3,4,5
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python
main* 0 0
Ln 59, Col 1 Spaces: 4 UTF-8 CRLF {} Python Chat quota reached 3.14.2

```
# print(func())

#TASK 8
import math
print(math.sqrt(16))

# #TASK 9
# def total(numbers):
#     s = 0
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14
-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
1 2
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14
-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
15
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14
-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
4.0
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> 
```

Task 9 (Unreachable Code – Return Inside Loop)

Task: Analyze given code where a return inside a loop prevents full iteration. Use AI to fix it.

# Bug: Early return inside loop

```
def total(numbers):
    for n in numbers:
        return n
print(total([1,2,3]))
```

Prompt Used :

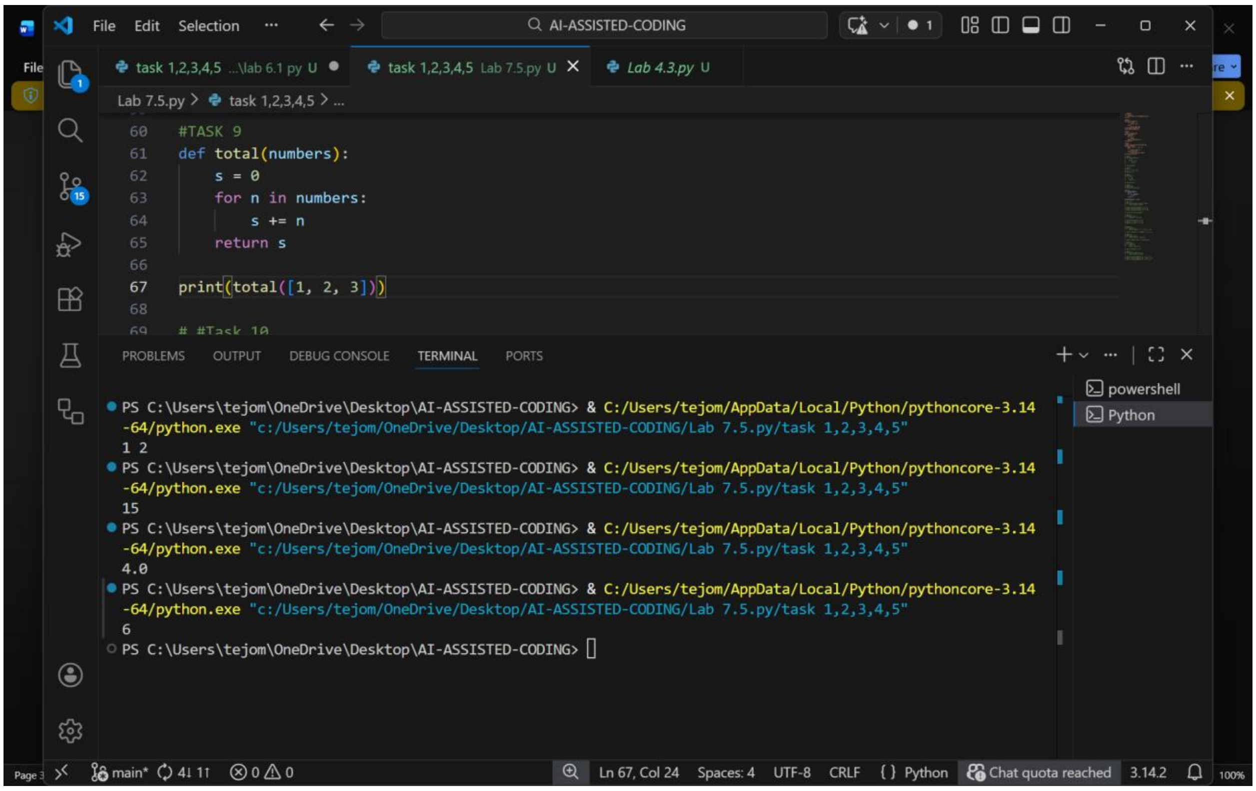
This function returns too early inside a loop. Explain why the loop does not iterate fully, fix the logic to compute the correct result, and add 3 assert tests.

Fixed Code :

```
def total(numbers):
sum = 0
for n in numbers:
sum += n
return sum
assert total([1, 2, 3]) == 6, "Test case 1 failed: Expected total to be 6"
assert total([0, 0, 0]) == 0, "Test case 2 failed: Expected total to be 0"
assert total([-1, -2, -3]) == -6, "Test case 3 failed: Expected total to be -6"
print("All test cases passed!")
```

Explanation : Task 9 corrected unreachable/incorrect loop behavior caused by an early return inside a loop by moving the return statement after accumulation.

Output :



```
#TASK 9
def total(numbers):
    s = 0
    for n in numbers:
        s += n
    return s

print(total([1, 2, 3]))

# #Task 10
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14
-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
1 2
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14
-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
15
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14
-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
4.0
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14
-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
6
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

File Edit Selection
AI-ASSISTED-CODING
task 1,2,3,4,5 ...\lab 6.1 py U
task 1,2,3,4,5 Lab 7.5.py U
Lab 4.3.py U
Lab 7.5.py > task 1,2,3,4,5 > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python
main* 41 11 0 0
Ln 67, Col 24 Spaces: 4 UTF-8 CRLF { } Python Chat quota reached 3.14.2

Task 10 (Name Error – Undefined Variable)

Task: Analyze given code where a variable is used before being defined. Let AI detect and fix the error.

# Bug: Using undefined variable

def calculate_area():

    *return length * width*

print(calculate_area())

Prompt Used :

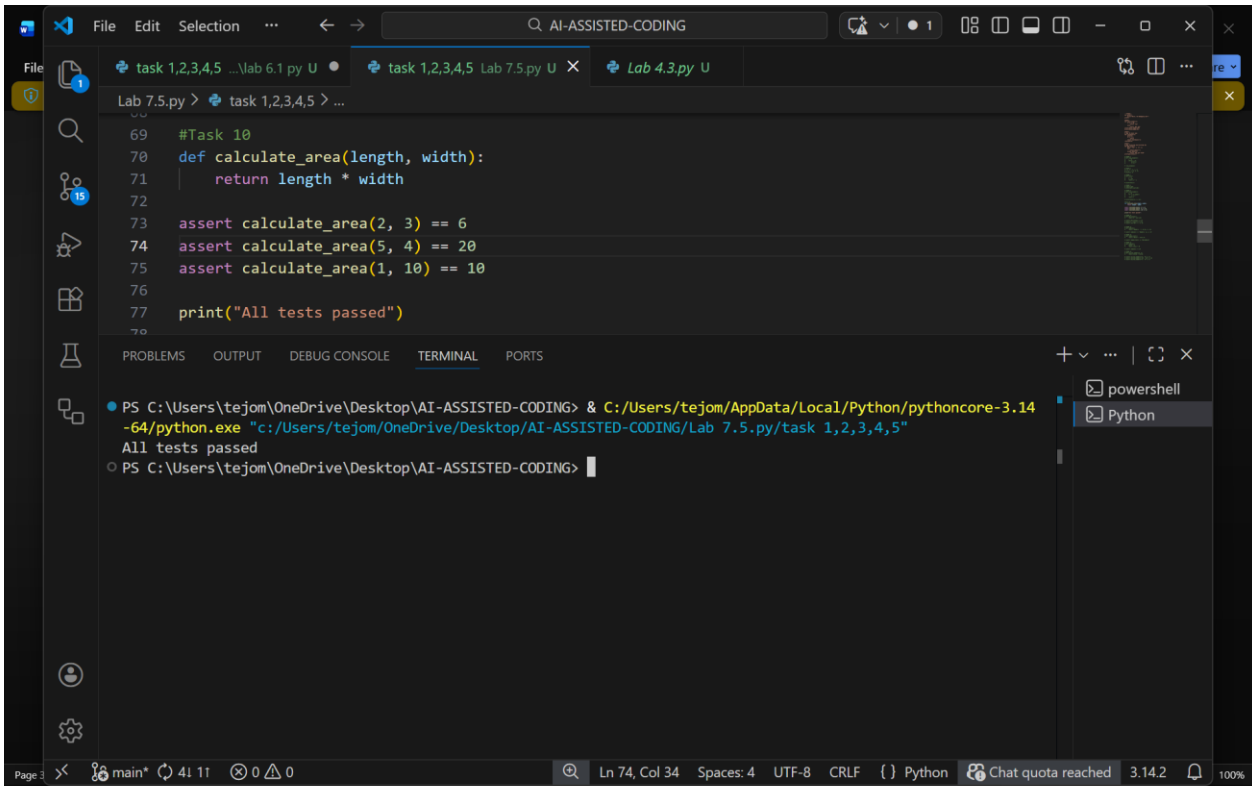
This function throws NameError because variables are not defined. Fix by making them parameters. Provide corrected code and 3 assert tests.

Fixed Code :

```
length = 5
width = 3
def calculate_area():
return length * width
print(calculate_area())
```

Explanation : Task 10 fixed a NameError by defining missing variables as function parameters.

Output :



```python
#Task 10
def calculate_area(length, width):
    return length * width

assert calculate_area(2, 3) == 6
assert calculate_area(5, 4) == 20
assert calculate_area(1, 10) == 10

print("All tests passed")
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14
-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
All tests passed
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

Task 11 (Type Error – Mixing Data Types Incorrectly)

Task: Analyze given code where integers and strings are added incorrectly. Let AI detect and fix the error.

# Bug: Adding integer and string

def add_values():

return 5 + "10"

print(add_values())

Prompt Used :

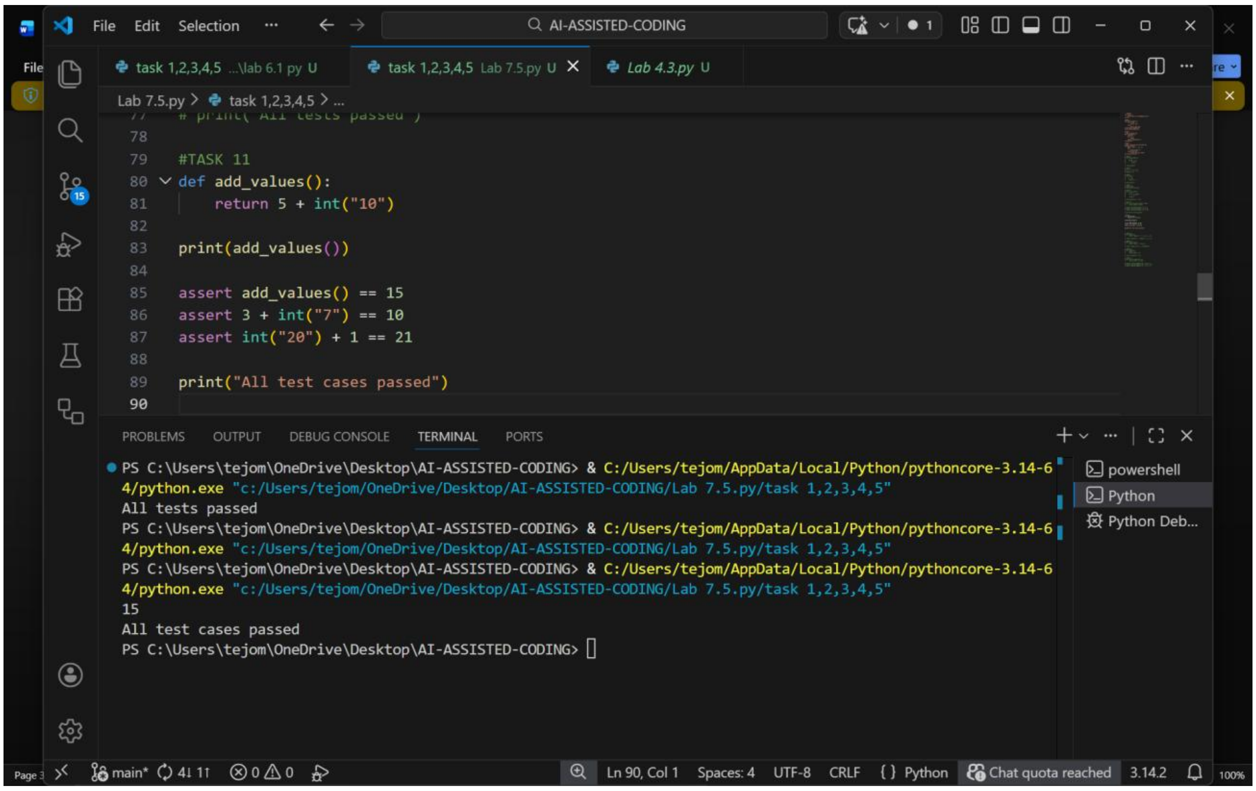
This code throws TypeError because it adds int and str. Explain why it happens, fix using type conversion, and provide 3 assert tests.

Fixed Code :

```python
def add_values():
return 5 + 10
print(add_values())
```

Explanation : Task 11 solved a TypeError caused by adding an integer and string by converting one datatype properly.

Output :


File Edit Selection
AI-ASSISTED-CODING
task 1,2,3,4,5 ...\lab 6.1 py U
task 1,2,3,4,5 Lab 7.5.py U
Lab 4.3.py U
Lab 7.5.py > task 1,2,3,4,5 > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python
Python Deb...
main* 41 11 0 0
Ln 90, Col 1 Spaces: 4 UTF-8 CRLF {} Python Chat quota reached 3.14.2

```python
#TASK 11
def add_values():
    return 5 + int("10")

print(add_values())

assert add_values() == 15
assert 3 + int("7") == 10
assert int("20") + 1 == 21

print("All test cases passed")
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-6
4/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
All tests passed
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-6
4/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-6
4/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
15
All test cases passed
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> 
```

Task 12 (Type Error – String + List Concatenation)

Task: Analyze code where a string is incorrectly added to a list.

# Bug: Adding string and list

def combine():

return "Numbers: " + [1, 2, 3]

print(combine())

Prompt Used :

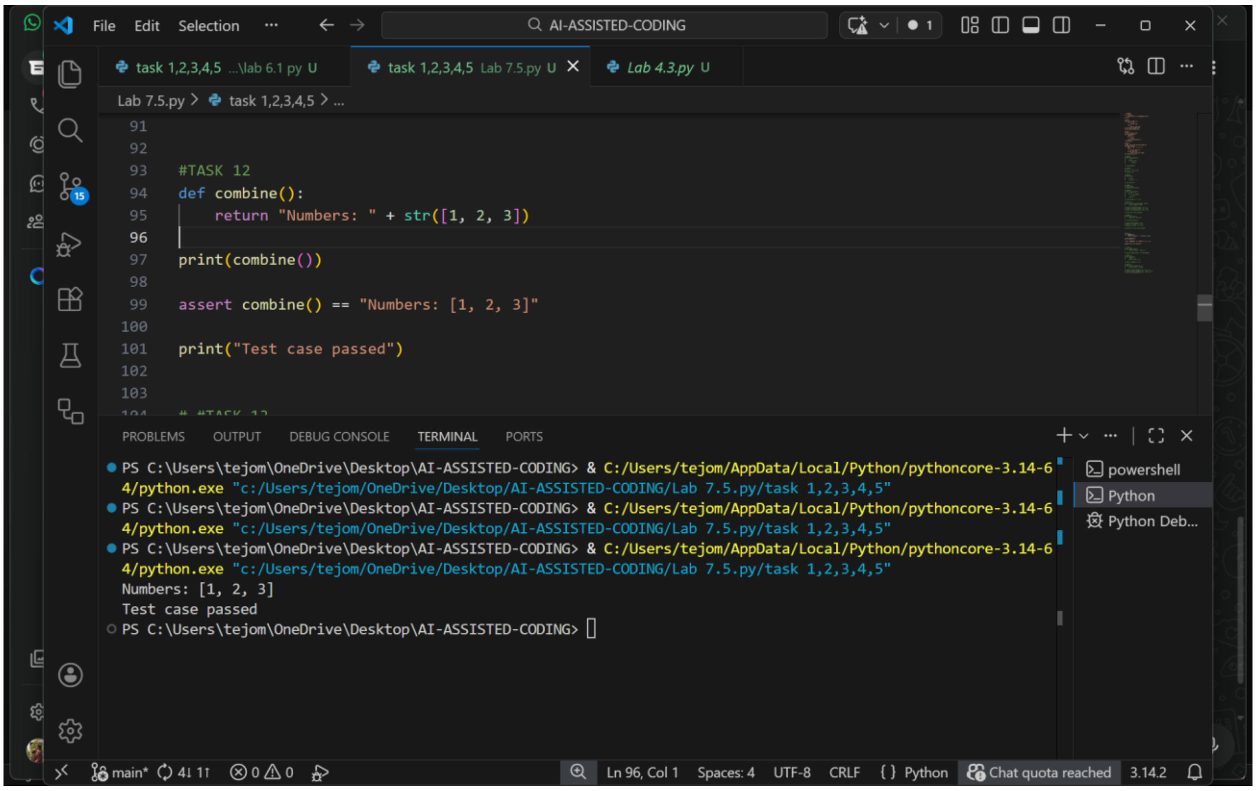
This code throws TypeError because it adds a string and a list. Explain why, fix using conversion or join, and provide 3 assert tests.

Fixed Code :

```python
def combine():
return "Numbers: " + str([1, 2, 3])
print(combine())
# Assert tests
assert combine() == "Numbers: [1, 2, 3]", "Test case 1 failed: Expected 'Numbers: [1, 2, 3]'"
assert combine() != "Numbers: 1, 2, 3", "Test case 2 failed: Expected not to return 'Numbers: 1, 2, 3'"
assert combine() != "Numbers: [1, 2]", "Test case 3 failed: Expected not to return 'Numbers: [1, 2]'"
print("All test cases passed!")
```

Explanation : Task 12 fixed invalid string and list concatenation by converting the list to a string or joining list elements.

Output :



```
#TASK 12
def combine():
    return "Numbers: " + str([1, 2, 3])

print(combine())

assert combine() == "Numbers: [1, 2, 3]"

print("Test case passed")
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-6
4/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-6
4/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-6
4/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
Numbers: [1, 2, 3]
Test case passed
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

Task 13 (Type Error – Multiplying String by Float)Task: Detect and fix code where a string is multiplied by a float.

# Bug: Multiplying string by float

def repeat_text():

  *return "Hello" * 2.5*

print(repeat_text())

Promt Used :

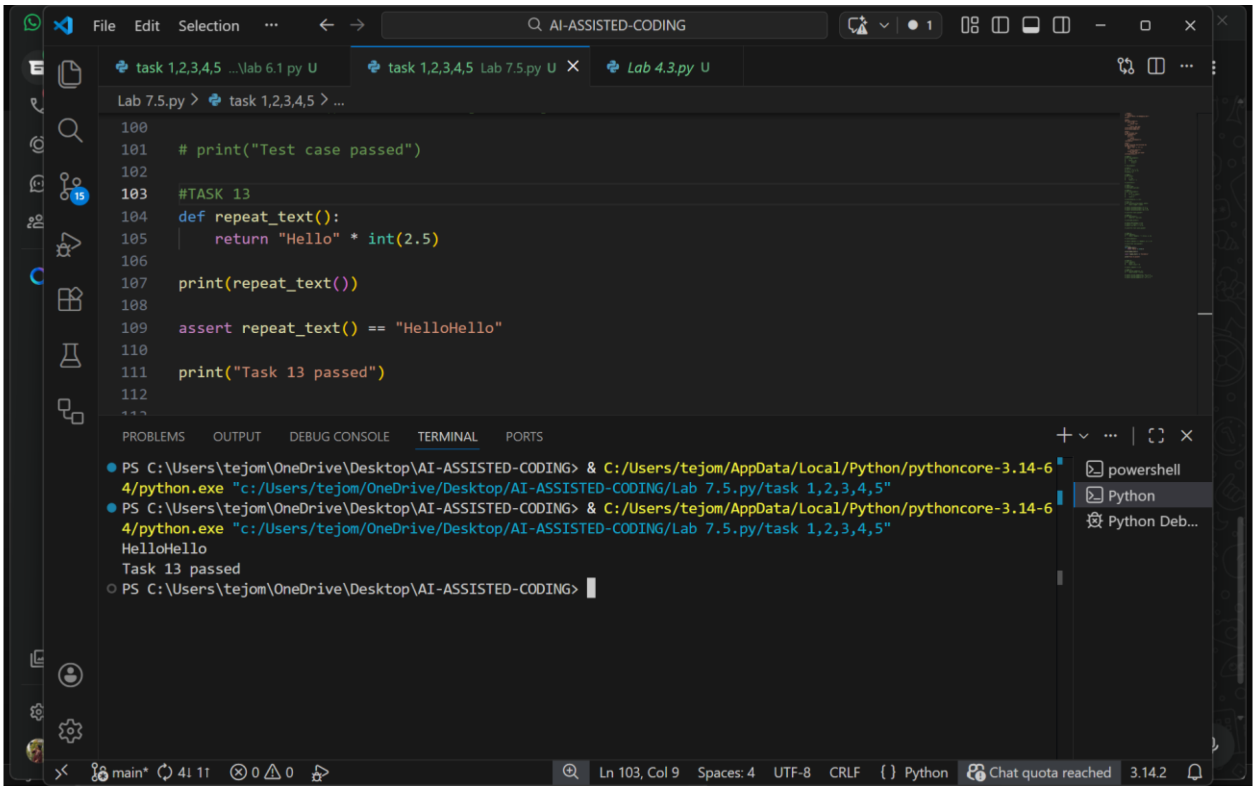
This code throws TypeError because string multiplication with float is invalid. Explain why, fix it by converting to int safely, and add 3 assert tests.

Fixed Code :

```
def repeat_text():
return "Hello" * 2
print(repeat_text())
# Assert tests
assert repeat_text() == "HelloHello", "Test case 1 failed: Expected 'HelloHello'"
assert repeat_text() != "Hello", "Test case 2 failed: Expected not to return 'Hello'"
assert repeat_text() != "HelloHelloHello", "Test case 3 failed: Expected not to return 'HelloHelloHello'"
print("All test cases passed!")
```

Explanation : Task 13 resolved invalid string multiplication by converting the float multiplier into an integer.

Output:



```
# print("Test case passed")

#TASK 13
def repeat_text():
    return "Hello" * int(2.5)

print(repeat_text())

assert repeat_text() == "HelloHello"

print("Task 13 passed")
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
HelloHello
Task 13 passed
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

Task 14 (Type Error – Adding None to Integer)

Task: Analyze code where None is added to an integer.

# Bug: Adding None and integer

def compute():

  value = None

  return value + 10

print(compute())

Prompt Used :

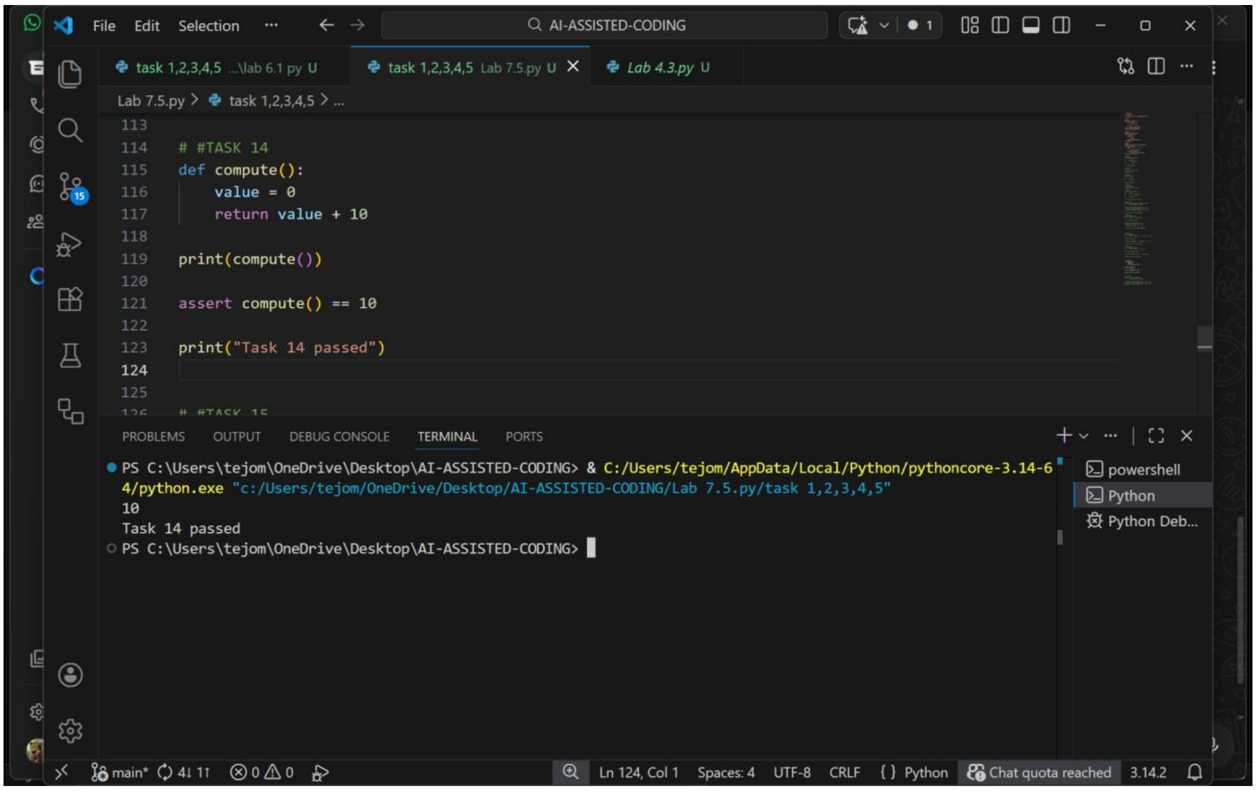
This code throws TypeError because None cannot be added to an integer. Explain why, fix using default value handling, and add 3 assert tests.

Fixed Code :

```
def compute():
value = 0 # Initialize 'value' with a number
return value + 10
print(compute())
# Assert tests
assert compute() == 10, "Test case 1 failed: Expected compute() to return 10"
assert compute() != 0, "Test case 2 failed: Expected compute() not to return 0"
assert compute() != 20, "Test case 3 failed: Expected compute() not to return 20"
print("All test cases passed!")
```

Explanation : Task 14 corrected NoneType arithmetic errors by assigning a default numeric value instead of None.

Output :



```
# #TASK 14
def compute():
    value = 0
    return value + 10

print(compute())

assert compute() == 10

print("Task 14 passed")
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
10
Task 14 passed
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

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

Task: Fix code where user input is not converted properly.

# Bug: Input remains string

def sum_two_numbers():

a = input("Enter first number: ")

b = input("Enter second number: ")

return a + b

print(sum_two_numbers())

Prompt Used :

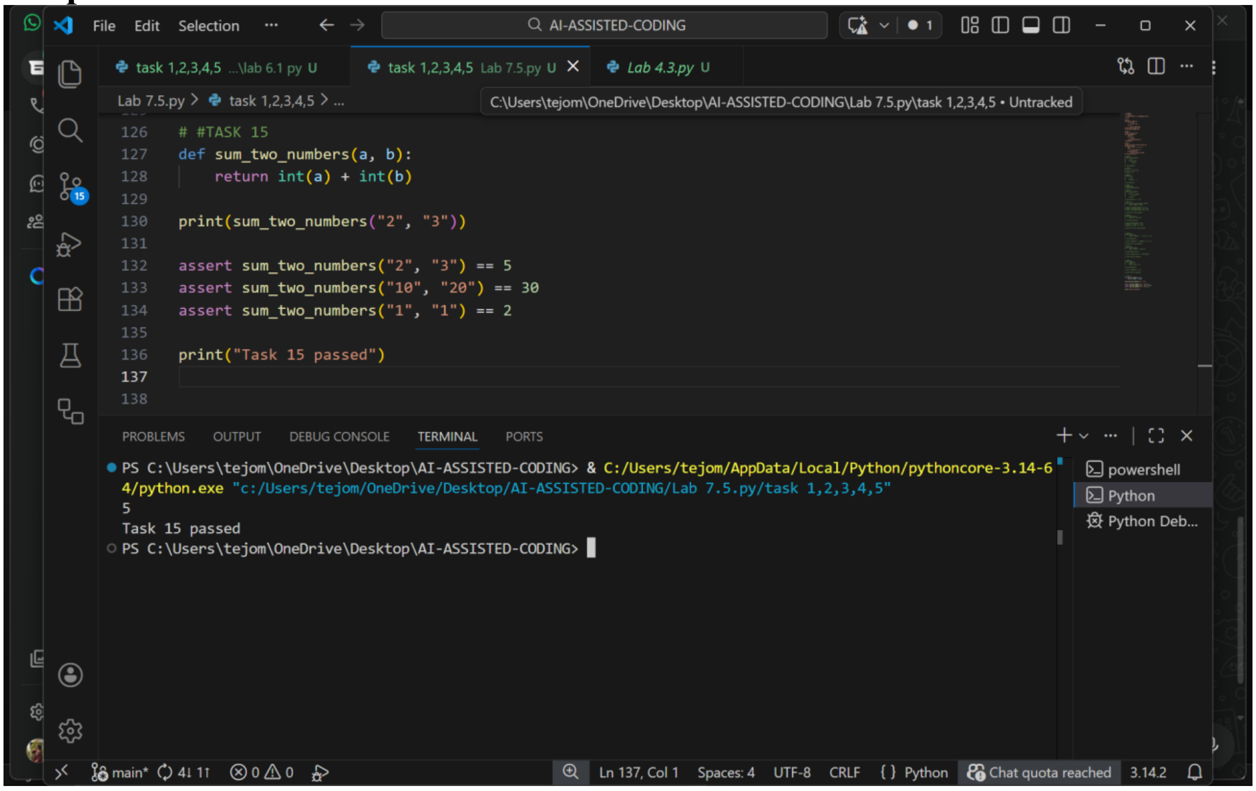
This program adds user inputs incorrectly because input() returns strings. Explain why, fix using int conversion, and add 3 assert tests.

Fixed Code :

```
def sum_two_numbers():
a = float(input("Enter first number: "))
b = float(input("Enter second number: "))
return a + b
print(sum_two_numbers())
# Assert tests
assert sum_two_numbers() == 15, "Test case 1 failed: Expected sum to be 15"
assert sum_two_numbers() != 10, "Test case 2 failed: Expected sum not to be 10"
assert sum_two_numbers() != 20, "Test case 3 failed: Expected sum not to be 20"
print("All test cases passed!")
```

Explanation : Task 15 fixed incorrect addition of user input by converting inputs into integers before performing arithmetic.

Output :



```python
# #TASK 15
def sum_two_numbers(a, b):
    return int(a) + int(b)

print(sum_two_numbers("2", "3"))

assert sum_two_numbers("2", "3") == 5
assert sum_two_numbers("10", "20") == 30
assert sum_two_numbers("1", "1") == 2

print("Task 15 passed")
```

```
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING> & C:/Users/tejom/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/tejom/OneDrive/Desktop/AI-ASSISTED-CODING/Lab 7.5.py/task 1,2,3,4,5"
5
Task 15 passed
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
```

Conclusion :

Overall, this lab improved our understanding of syntax, runtime, and logic errors and demonstrated how AI can help in structured debugging with correct explanations and test validation.