

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 :

The screenshot shows a code editor interface with several tabs open. The main tab contains Python code for a task involving floating-point numbers. A sidebar on the right provides a solution and key points about the use of `math.isclose()`.

```

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

def compare_floats(a, b, tolerance=1e-9):
    """Compare floats using absolute difference tolerance"""
    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.isclose"
assert compare_floats(0.7 + 0.1, 0.8, tolerance=1e-9), "Tolerance method"
assert compare_floats_isclose(0.7 + 0.1, 0.8), "math.isclose"

print("All tests passed!")

```

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

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

```

import math

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

def compare_floats_isclose(a, b):
    """Compare floats using math.isclose"""
    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.isclose"
assert compare_floats(0.7 + 0.1, 0.8, tolerance=1e-9), "Tolerance method"
assert compare_floats_isclose(0.7 + 0.1, 0.8), "math.isclose"

print("All tests passed!")

```

Key Points

- `math.isclose(a, b)` - Uses relative a absolute tolerance (recommended for floating-point precision issues)

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 :

```

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 :

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows a tree view of files under "AI-ASSISTED-CODING". Files include Lab 7.5.py, task 1,2,3,4,5, Lab 6.1.py, Lab 1 Task1,2,3,4,5.py, lab 6.1 py, Lab 4.3.py, Task 1,2,3,4,5.py, LAB2.PY, task 3.py, Task 1.py, task 4.py, task 2.py, lab4.1.py, customer-email classification, internet-classification, lab5 .py, Task 1.py ass-3, Task 2,3,4,5,6.py ass-3, users.txt.
- EDITOR:** Displays code for "task 1,2,3,4,5 > task 1,2,3,4,5 > ...".

```
1 #TASK 1
2 def greet():
3     print("Hello, AI Debugging Lab!")
4     greet()
5
6 #TASK 2
7 def check_number(n):
8     if n == 10:
9         return "Ten"
10    else:
11        return "Not Ten"
12 print(check_number(10))
13 print(check_number(5))
14
15 #TASK 3
16 # Bug: No base case
17 def countdown(n):
18     print(n)
19     if n == 0:
20         return
21     return countdown(n-1)
22 countdown(5)
```
- TERMINAL:** Shows the command "python.exe" running and printing numbers 5, 4, 3, 2, 1, 0.
- SESSIONS:** Lists recent tasks:
 - Task: Analyze given code where a mutable ... Failed after 2s. 8 mins ago
 - Task: Analyze given code where a mutable ... Failed after 3s. 9 mins ago
 - Fixing infinite recursion with base case and... Completed in 3s. 16 mins ago
- CHAT:** Shows a message: "You've reached the limit for chat mes... Upgrade".

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 :

```
File Edit Selection View Go ... < > Q 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 ... CHAT + ... SESSIONS G SSESSIONS ... 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 MORE (2)
```

```
EXPLORER AI-ASSISTED-CODING Lab 7.5.py > task 1,2,3,4,5 > get_value
7 def check_number(n):
8     if n == 10:
9         return "Not Ten"
10    print(check_number(10))
11    print(check_number(5))
12
13 #TASK 3
14 # Bug: No base case
15 def countdown(n):
16     print(n)
17     if n == 0:
18         return
19     return countdown(n-1)
20
21 countdown(5)
22
23
24 #TASK 4
25 # Bug: Accessing non-existing key
26 def get_value():
27     data = {"a": 1, "b": 2}
28     try:
29         return data["c"]
30     except KeyError:
31         return "Key not found"
32
33 print(get_value())
34
35
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | X
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
Python Python
2
1
0
Key not found
PS C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>
Ln 31, Col 31 Spaces: 4 UTF-8 CRLF { } Python Chat quota reached 3.14.2
```

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

task 1,2,3,4,5

Describe what to build next

Auto CRLF

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 :

The screenshot shows the VS Code interface with the AI-Assisted Coding extension active. The code editor displays a Python file named `task 1,2,3,4,5` containing the following code:

```
def get_value():
    data = {"a": 1, "b": 2}
    try:
        return data["c"]
    except KeyError:
        return "Key not found"
print(get_value())'''
```

The AI panel on the right provides a suggestion to use `get()` instead of direct indexing to handle missing keys. The terminal below shows the output of running the script, which prints "Key not found".

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 :

```
defloop_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 :

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows three files: "task 1,2,3,4,5 ...\\lab 6.1.py", "task 1,2,3,4,5 Lab 7.5.py", and "Lab 4.3.py".
- Terminal:** Displays the command "python.exe" being run with the file "Lab 7.5.py". The output shows the values 1 and 2 printed to the console.
- Status Bar:** Shows the current file path as "C:\\Users\\tejom\\OneDrive\\Desktop\\AI-ASSISTED-CODING\\Lab 7.5.py", the line number as "Ln 45, Col 20", and the character offset as "Spaces: 4". It also indicates "Python" as the language and "Chat quota reached".

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 :

The screenshot shows the Visual Studio Code interface. The code editor displays a Python script named 'Lab 7.5.py' with the following content:

```

39     #         i += 1
40
41     # loop_example()
42
43     #TASK 6
44     # a, b = (1, 2, 3)
45     a, b, _ = [1, 2, 3]
46     print(a, b)
47
48     # #TASK 7

```

The terminal tab shows the output of running the script:

```

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

```

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():

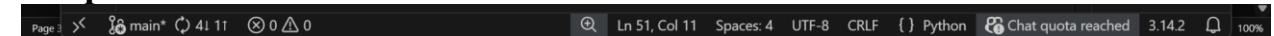
```

```
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 :

A screenshot of a code editor window. The status bar at the bottom shows: Page 3, main*, 4111, 0, 3.14.2, 100%. Other icons include a magnifying glass, a file icon, and a refresh icon.

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 :

The screenshot shows the Visual Studio Code interface. The top bar has tabs for 'File', 'Edit', 'Selection', etc., and a search bar 'AI-ASSISTED-CODING'. The main area shows three tabs: 'task 1,2,3,4,5 ...Lab 6.1.py' (marked with a yellow warning icon), 'task 1,2,3,4,5 Lab 7.5.py' (marked with a red error icon), and 'Lab 4.3.py'. Below the tabs is a code editor with the following Python code:

```
53
54     # print(func())
55
56     #TASK 8
57     import math
58     print(math.sqrt(16))
59
60     # #TASK 9
61     # def total(numbers):
62     #     s = 0
```

The terminal below shows the execution of the code:

```
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>
```

The status bar at the bottom indicates the file is 'main*' and the line is 'Ln 59, Col 1'. It also shows 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', 'Chat quota reached', '3.14.2', and '100%'. On the left, there are icons for file operations like save, open, and delete, and a sidebar with a tree view.

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 :

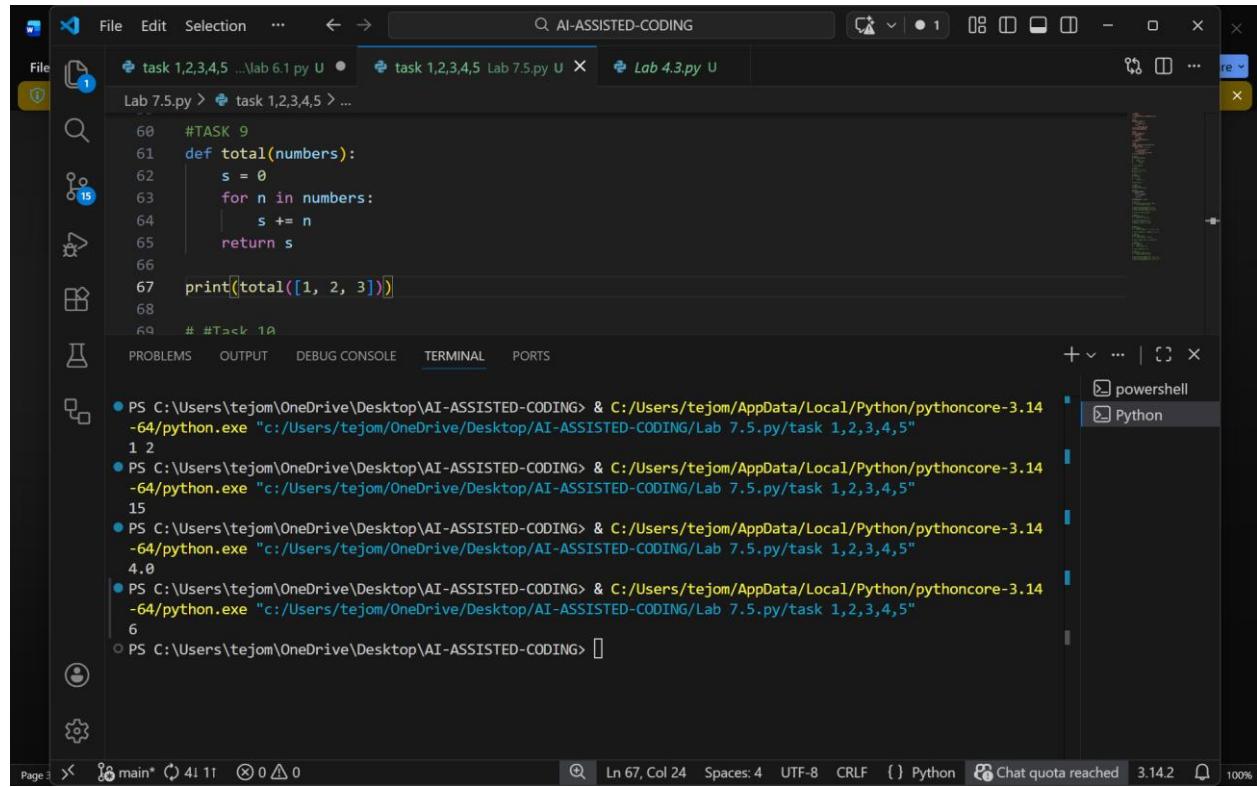
This function returns too early inside a loop. Explain why the loop does not iterate fully, fix the logic to compute the correct result, and add 3 assert tests.

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 :



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows files: task 1,2,3,4,5 ...\\lab 6.1.py, Lab 7.5.py, and Lab 4.3.py.
- Code Editor:** Displays Python code:

```
60 #TASK 9
61 def total(numbers):
62     s = 0
63     for n in numbers:
64         s += n
65     return s
66
67 print(total([1, 2, 3]))
```
- Terminal:** Shows command-line output for four different Python environments (PS, C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING>), each running Lab 7.5.py and printing the value 6.
- Output:** Shows a list of tasks: powershell and Python.
- Status Bar:** Includes file path (C:\Users\tejom\OneDrive\Desktop\AI-ASSISTED-CODING\Lab 7.5.py), line (Ln 67, Col 24), spaces (Spaces: 4), encoding (UTF-8), CRLF, Python, Chat quota reached, version (3.14.2), and zoom (100%).

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 :

```
#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 :

```
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 :

The screenshot shows a code editor interface with three tabs open: 'task 1,2,3,4,5 ...lab 6.1.py U', 'task 1,2,3,4,5 Lab 7.5.py U', and 'Lab 4.3.py U'. The 'Lab 7.5.py' tab contains the following Python code:

```

78     #TASK 11
80     def add_values():
81         return 5 + int("10")
82
83     print(add_values())
84
85     assert add_values() == 15
86     assert 3 + int("7") == 10
87     assert int("20") + 1 == 21
88
89     print("All test cases passed")
90

```

The terminal below shows the execution of the script and its output:

```

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 :

```

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 :

```

91
92
93 #TASK 12
94 def combine():
95     return "Numbers: " + str([1, 2, 3])
96
97 print(combine())
98
99 assert combine() == "Numbers: [1, 2, 3]"
100
101 print("Test case passed")
102
103

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- 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"
- 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:

The screenshot shows the Visual Studio Code interface. The code editor has three tabs open: 'task 1,2,3,4,5 ...Lab 6.1.py' (inactive), 'task 1,2,3,4,5 Lab 7.5.py' (active), and 'Lab 4.3.py'. The active tab contains Python code for Task 13:

```

100
101     # print("Test case passed")
102
103 #TASK 13
104 def repeat_text():
105     return "Hello" * int(2.5)
106
107 print(repeat_text())
108
109 assert repeat_text() == "HelloHello"
110
111 print("Task 13 passed")
112

```

The terminal below shows the execution of the script:

```

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"
HelloHello
Task 13 passed

```

The status bar at the bottom indicates the file is main+, line 103, column 9, spaces: 4, and encoding is UTF-8.

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 :

The screenshot shows a dark-themed instance of Visual Studio Code. In the center-left, there's a code editor with several tabs open. The active tab contains Python code for Task 14:

```
113 # #TASK 14
114 def compute():
115     value = 0
116     return value + 10
117
118 print(compute())
119
120 assert compute() == 10
121
122 print("Task 14 passed")
123
124
125 # #TASK 15
```

Below the code editor is a terminal window showing command-line output:

```
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>
```

The right side of the interface features a sidebar with various icons, including one for Python. A status bar at the bottom provides information about the current file and workspace.

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 :

The screenshot shows the Visual Studio Code interface. The left sidebar has icons for file operations like Open, Save, Find, and Cut/Paste. The main editor area contains Python code for Task 15:

```
126 # #TASK 15
127 def sum_two_numbers(a, b):
128     return int(a) + int(b)
129
130 print(sum_two_numbers("2", "3"))
131
132 assert sum_two_numbers("2", "3") == 5
133 assert sum_two_numbers("10", "20") == 30
134 assert sum_two_numbers("1", "1") == 2
135
136 print("Task 15 passed")
137
138
```

The terminal tab at the bottom shows the output of running the script:

```
● 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>
```

The status bar at the bottom right indicates the current line (Ln 137), column (Col 1), and other settings like spaces per tab (Spaces: 4) and encoding (UTF-8).

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.