

LAB ASSIGNMENT – 7.5

NAME – RAMYASRI

2303A510i9

BATCH – 30

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=[]):
```

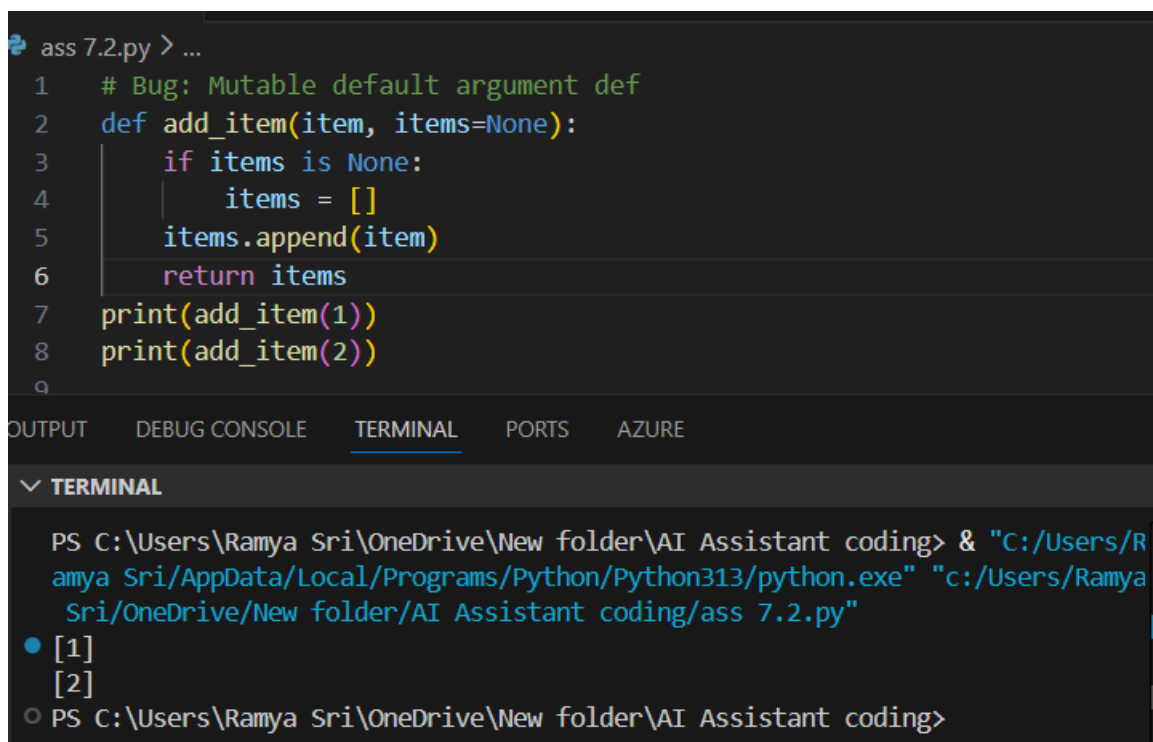
```
    items.append(item)
```

```
return items
```

```
print(add_item(1))
```

```
print(add_item(2))
```

Expected Output: Corrected function avoids shared list bug.



The screenshot shows a code editor with the following Python code:

```
1 # Bug: Mutable default argument def
2 def add_item(item, items=None):
3     if items is None:
4         items = []
5     items.append(item)
6     return items
7 print(add_item(1))
8 print(add_item(2))
9
```

Below the code editor is a terminal window with the following output:

```
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & "C:/Users/Ramya Sri/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Ramya Sri/OneDrive/New folder/AI Assistant coding/ass 7.2.py"
[1]
[2]
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

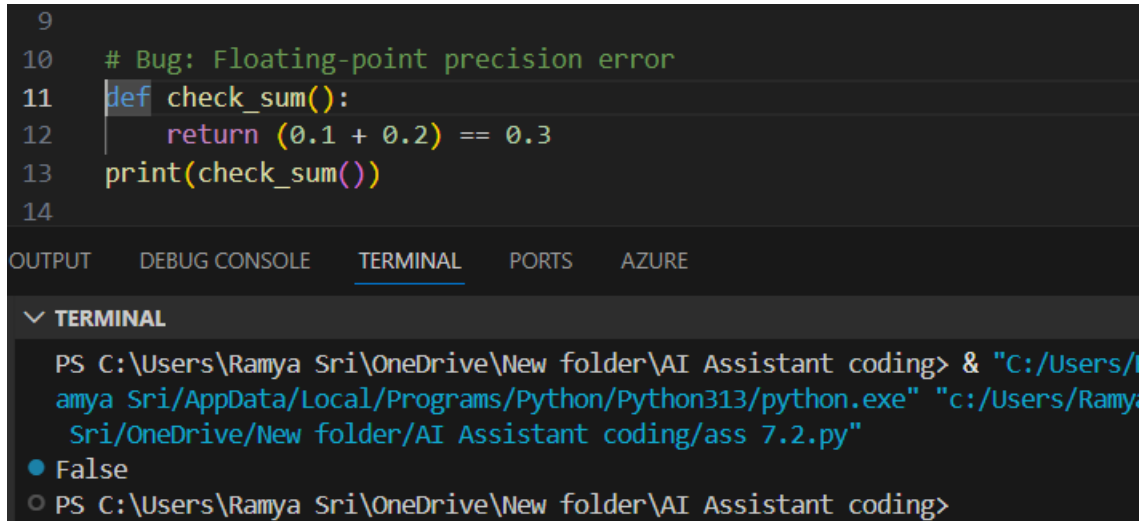
Task 2 (Floating-Point Precision Error)

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())
```

Expected Output: Corrected function



The screenshot shows a code editor with a dark theme. The code is as follows:

```
9  
10 # Bug: Floating-point precision error  
11 def check_sum():  
12     return (0.1 + 0.2) == 0.3  
13 print(check_sum())  
14
```

Below the code editor, there are tabs for OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and AZURE. The TERMINAL tab is selected, showing the following command and output:

```
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & "C:/Users/Ramya Sri/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Ramya Sri/OneDrive/New folder/AI Assistant coding/ass 7.2.py"  
● False  
○ PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant 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)
```

Expected Output : Correct recursion with stopping condition.

```
15 # Bug: No base case def
16 def countdown(n):
17     if n == 0:
18         return
19     print(n)
20     return countdown(n-1)
21 countdown(5)
22
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

▼ TERMINAL

```
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & "C:/Users/Ramya Sri/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Ramya Sri/OneDrive/New folder/AI Assistant coding/ass 7.2.py"
5
4
3
2
1
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

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())
```

Expected Output: Corrected with .get() or error handling.

```
3 # Bug: Accessing non-existing key def
4 def get_value():
5     data = {"a": 1, "b": 2}
6     return data.get("c", None)
7 print(get_value())
8
```

OUTPUT DEBUG CONSOLE **TERMINAL** PORTS AZURE

TERMINAL

```
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & "C:/Users/R
amy Sri/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Ramya
Sri/OneDrive\New folder\AI Assistant coding/ass 7.2.py"
None
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

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)

Expected Output: Corrected loop increments i.

```
27 # Bug: Infinite loop
28 def loop_example():
29     i = 0
30     while i < 5:
31         print(i)
32         i += 1
33
```

PROBLEMS OUTPUT **TERMINAL** PORTS

TERMINAL

```
PS C:\Users\Shree priya\Downloads> & 'c:\Users\Shree
s-python.debugpy-2025.18.0-win32-x64\bundled\libs\debu
4.0
```

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)

Expected Output: Correct unpacking or using _ for extra values.

```
34 # Bug: Wrong unpacking
35 a, b, c = (1, 2, 3)
36
```

PROBLEMS OUTPUT **TERMINAL** PORTS

▼ **TERMINAL**

```
PS C:\Users\Shree priya\Downloads> & 'c:\Users\
s-python.debugpy-2025.18.0-win32-x64\bundled\li
4.0
6
```

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
```

Expected Output : Consistent indentation applied

```
37 # Bug: Mixed indentation
38 def func():
39     x = 5
40     y = 10
41     return x+y
42
```

PROBLEMS OUTPUT **TERMINAL** PORTS

▼ **TERMINAL**

```
PS C:\Users\Shree priya\Downloads>
s-python.debugpy-2025.18.0-win32-
4.0
6
50
15
```

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))
```

Expected Output: Corrected to import math

```
43 # Bug: Wrong import
44 import math
45 print(math.sqrt(16))
46
```

PROBLEMS OUTPUT **TERMINAL** PORTS

▼ **TERMINAL**

```
PS C:\Users\Shree priya\Downloads> & 'c:\Users\Shree priya\AppData\Local\Microsoft\WindowsApps\python.debugpy-2025.18.0-win32-x64\bundled\libs\python.exe'
4.0
6
50
15
```

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]))
```

Expected Output: Corrected code accumulates sum and returns after loop.

```
47 # Bug: Early return inside loop
48 def total(numbers):
49     sum_total = 0
50     for n in numbers:
51         sum_total += n
52     return sum_total
53 print(total([1,2,3]))
54
```

PROBLEMS OUTPUT **TERMINAL** PORTS

▼ **TERMINAL**

```
PS C:\Users\Shree priya\Downloads> & 'c:\Users\Shree priya\AppData\Local\Microsoft\WindowsApps\python.debugpy-2025.18.0-win32-x64\bundled\libs\python.exe'
4.0
6
50
15
Numbers: [1, 2, 3]
```

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())
```

Requirements:

- Run the code to observe the error.

- Ask AI to identify the missing variable definition.
- Fix the bug by defining length and width as parameters.
- Add 3 assert test cases for correctness.

Expected Output :

- Corrected code with parameters.
- AI explanation of the bug.

Successful execution of assertions.



The screenshot shows a code editor with the following Python code:

```

55 # Bug: Using undefined variable
56 def calculate_area(length, width):
57     return length * width
58 print(calculate_area(5, 10))
59

```

Below the code editor, the terminal output is displayed:

```

PS C:\Users\Shree priya\Downloads> & 'c:\Users\Shree priya\AppData\Local\Microsoft\WindowsApps\python.debugpy-2025.18.0-win32-x64\bundled\libs\python.exe'
4.0
6
50
15

```

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())

```

Requirements:

- Run the code to observe the error.
- AI should explain why int + str is invalid.
- Fix the code by type conversion (e.g., int("10") or str(5)).
- Verify with 3 assert cases.

Expected Output #6:

- Corrected code with type handling.
- AI explanation of the fix.

Successful test validation.

```
50
51 # Bug: Adding integer and string
52 def add_values():
53     return 5 + int("10")
54     print(add_values())
55
56
57 PROBLEMS OUTPUT TERMINAL PORTS
58
59 > TERMINAL
60
61 PS C:\Users\Shree priya\Downloads> & 'c:\Users\Shree priya\Downloads\python-debugpy-2025.18.0-win32-x64\bundle
4.0
6
50
15
```

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())
```

Requirements:

- Run the code to observe the error.
- Explain why str + list is invalid.
- Fix using conversion (str([1,2,3]) or " ".join()).
- Verify with 3 assert cases.

Expected Output:

- Corrected code
- Explanation

Successful test validation

```
65 # Bug: Adding string and list
66 def combine():
67     return "Numbers: " + str([1, 2, 3])
68     print(combine())
69
70
71 PROBLEMS OUTPUT TERMINAL PORTS
72
73 > TERMINAL
74
75 PS C:\Users\Shree priya\Downloads> & 'c:\Users\Shree priya\Downloads\python-debugpy-2025.18.0-win32-x64\bundle
4.0
6
50
15
Numbers: [1, 2, 3]
```

Task 13 (Type Error – Multiplying String by Float)

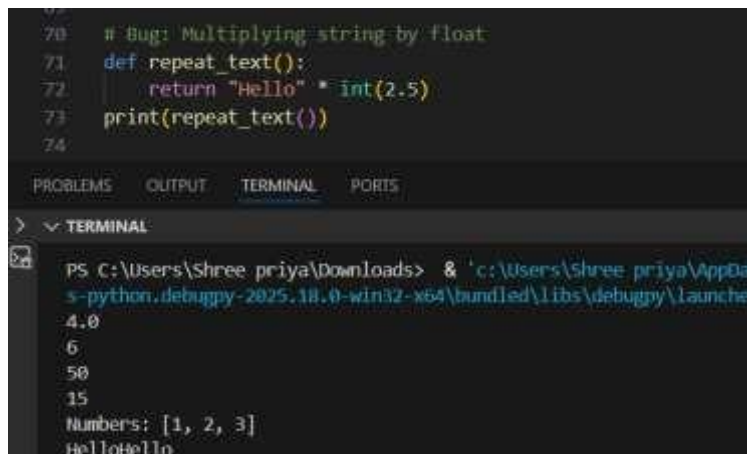
Task: Detect and fix code where a string is multiplied by a float. # Bug: Multiplying string by float

```
def repeat_text():    return "Hello" * 2.5    print(repeat_text())
```

Requirements:

- Observe the error.
- Explain why float multiplication is invalid for strings.
- Fix by converting float to int.

Add 3 assert test cases



The screenshot shows a code editor with a Python function `repeat_text()` that returns `"Hello" * int(2.5)` and prints the result. Below the code, a terminal window displays the output of the function, which is `HelloHello`. The terminal also shows some other output, including `4.0`, `6`, `50`, `15`, and `Numbers: [1, 2, 3]`.

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())
```

Requirements:

- Run and identify the error.
- Explain why `NoneType` cannot be added.
- Fix by assigning a default value.
- Validate using asserts.

```
75 # Bug: Adding None and integer
76 def compute():
77     value = 0
78     return value + 10
79 print(compute())
80
81
```

PROBLEMS OUTPUT TERMINAL PORTS

▼ TERMINAL

```
PS C:\Users\Shree priya\Downloads> & 'c:\Users\Shree priya\AppData\Local\Microsoft\WindowsApps\python.debugpy-2025.18.0-win32-x64\python.exe'
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10
```

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())
```

Requirements:

- Explain why input is always string.
- Fix using int() conversion.

```
86 # Bug: input remains string
87 def sum_two_numbers():
88     a = int(input("Enter first number: "))
89     b = int(input("Enter second number: "))
90     return a + b
91
92 print(sum_two_numbers())
93
```

PROBLEMS OUTPUT TERMINAL PORTS

✓ TERMINAL

```
PS C:\Users\Shree priya\Downloads> .& 'c:\Users\Shree priya\AppData
s-python.debugpy.2025.18.0-win32-x64\bundled\libs\debugpy\launcher
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10
Enter first number: 8
Enter second number: 6
14
```