

ASSIGNMENT-7.5

M.AKHIL REDDY

2303A52315

BATCH-45

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 list `items` is **shared across all function calls**
- That's why the second call keeps data from the first call

The screenshot shows a dark-themed interface of the Visual Studio Code editor. On the left is a sidebar with icons for file operations like Open, Save, Find, and others. The main area has a tab bar with '7.5.py' and an 'X'. Below the tabs is a code editor with the following Python code:

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

Below the code editor is a navigation bar with links: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing the command line output:

```
live/Desktop/akhils ai coding/7.5.py"
[1]
[2]
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

Task 2 (Floating-Point Precision Error)

Task2: 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

Floating-point numbers are stored in **binary**, so some decimal values cannot be represented exactly.

```
File Edit Selection View Go ... ⏪ ⏩ akhils ai coding
7.5.py x
7.5.py > ...
1 # def add_item(item, items=None):
2 #     if items is None:
3 #         items = []
4 #     items.append(item)
5 #     return items
6
7 # print(add_item(1))
8 # print(add_item(2))
9
10 def check_sum():
11     return abs((0.1 + 0.2) - 0.3) < 1e-9
12
13 print(check_sum())
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe/Desktop/akhils ai coding/7.5.py"

True

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>

Ln 14, Col 1 Spaces: 4 UTF-8

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.

There is **no base case**, so the function keeps calling itself forever.

This leads to:

- Infinite recursion
- RecursionError: maximum recursion depth exceeded

The screenshot shows a dark-themed instance of Visual Studio Code. In the center-left pane, there is a code editor with a file named "7.5.py". The code contains a function "countdown(n)" that prints integers from n down to 0. Lines 22 and 23 show the call "countdown(5)". The terminal at the bottom displays the output of running the script: "2", "1", and "0". The status bar at the bottom right indicates "In 23, Col 1".

```
7.5.py
7.5.py > ...
7  # print(add_item(1))
8  # print(add_item(2))
9
10 # def check_sum():
11 #     return abs((0.1 + 0.2) - 0.3) < 1e-9
12
13 # print(check_sum())
14
15 def countdown(n):
16     if n == 0:
17         print(0)
18         return
19     print(n)
20     countdown(n - 1)
21
22 countdown(5)
23
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

2
1
0
PS C:\Users\achyu\OneDrive\Desktop\akhils ai 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.

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows icons for file operations like Open, Save, Find, and Refresh. The main editor area has tabs for '7.5.py' and '7.5.py > ...'. The code in '7.5.py' is:

```
16     #     if n == 0:
17     #         print(0)
18     #         return
19     #     print(n)
20     #     countdown(n - 1)
21
22 # countdown(5)
23
24 def get_value():
25     data = {"a": 1, "b": 2}
26     try:
27         return data["c"]
28     except KeyError:
29         return "Key not found"
30
31 print(get_value())
32
33
34
```

The terminal at the bottom shows the command being run and its output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/7.5.py"
Key not found
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

given very well

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.

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar contains icons for file operations like Open, Save, Find, and others. The main editor area shows a Python script named '7.5.py' with the following code:

```
24 # def get_value():
25 #     data = {"a": 1, "b": 2}
26 #     try:
27 #         return data["c"]
28 #     except KeyError:
29 #         return "Key not found"
30
31 # print(get_value())
32
33 def loop_example():
34     i = 0
35     while i < 5:
36         print(i)
37         i += 1
38
39 loop_example()
40
41
42
43
```

The terminal at the bottom shows the command line prompt 'PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>' followed by the numbers 2, 3, and 4 on separate lines.

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.

The screenshot shows a dark-themed instance of Visual Studio Code. A file named `7.5.py` is open in the editor. The code contains several lines of Python, including a function definition and a loop. The terminal below the editor shows the command `python 7.5.py` being run and its output, which is the numbers 1 and 2 on separate lines.

```
27     #         return data["c"]
28     #     except KeyError:
29     #         return "Key not found"
30
31     # print(get_value())
32
33     # def loop_example():
34     #     i = 0
35     #     while i < 5:
36     #         print(i)
37     #         i += 1
38
39     # loop_example()
40
41 a, b, _ = (1, 2, 3)
42 print(a, b)
43
44
45
46
47
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/7.5.py"
1
2
PS C:\Users\achyu\OneDrive\Desktop\akhils ai 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
```

Expected Output : Consistent indentation applied.

```
# i = 0
# while i < 5:
#     print(i)
#     i += 1
#
# loop_example()
#
# a, b, _ = (1, 2, 3)
# print(a, b)
#
def func():
    x = 5
    y = 10
    return x + y
#
print(func())

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/7.5.py"

15

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>

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

```
# i = 0
# while i < 5:
#     print(i)
#     i += 1
#
# loop_example()
#
# a, b, _ = (1, 2, 3)
# print(a, b)
#
def func():
    x = 5
    y = 10
    return x + y
#
# print(func())
#
import math
print(math.sqrt(16))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/7.5.py"

4.0

PS C:\Users\achyu\OneDrive\Desktop\akhils ai 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]))
```

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

```
File Edit Selection View Go ... ⏪ ⏩ Q akhils ai coding
 7.5.py x
 7.5.py > ...
45      #     x = 5
46      #     y = 10
47      #     return x + y
48
49      # print(func())
50
51      # import math
52      # print(math.sqrt(16))
53
54  def total(numbers):
55      s = 0
56      for n in numbers:
57          s += n
58      return s
59
60  print(total([1, 2, 3]))
61
62
63
64
65
66

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/7.5.py"
6
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

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.

```

49 # print(func())
50
51 # import math
52 # print(math.sqrt(16))
53
54 # def total(numbers):
55 #     s = 0
56 #     for n in numbers:
57 #         s += n
58 #     return s
59
60 # print(total([1, 2, 3]))
61
62 def calculate_area(length, width):
63     return length * width
64
65 assert calculate_area(5, 4) == 20
66 assert calculate_area(10, 2) == 20
67 assert calculate_area(3, 3) == 9
68
69 print("All test cases passed")
70

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/Desktop/akhils ai coding/7.5.py"

All test cases passed

PS C:\Users\achyu\OneDrive\Desktop\akhils ai 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())
```

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.

```

7.5.py
1  #!/usr/bin/python
2
3  def calculate_area(length, width):
4      return length * width
5
6  # assert calculate_area(5, 4) == 20
7  # assert calculate_area(10, 2) == 20
8  # assert calculate_area(3, 3) == 9
9
10 # print("All test cases passed")
11
12 def add_values():
13     return 5 + int("10")
14
15 assert add_values() == 15
16 assert 3 + int("7") == 10
17 assert int("4") + 6 == 10
18
19 print("All test cases passed")
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84

```

TERMINAL

```

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/7.5.py"
All test cases passed
PS C:\Users\achyu\OneDrive\Desktop\akhils ai 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())
```

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

```

File Edit Selection View Go ... ⏪ akhils ai coding ⏴ 08
7.5.py X
7.5.py > ...
66 # assert calculate_area(10, 2) == 20
67 # assert calculate_area(3, 3) == 9
68
69 # print("All test cases passed")
70
71 # def add_values():
72 #     return 5 + int("10")
73
74 # assert add_values() == 15
75 # assert 3 + int("7") == 10
76 # assert int("4") + 6 == 10
77
78 # print("All test cases passed")
79
80 def combine():
81     return "Numbers: " + str([1, 2, 3])
82
83 assert combine() == "Numbers: [1, 2, 3]"
84 assert "A" + str([1]) == "A[1]"
85 assert "List: " + str([]) == "List: []"
86
87 print("All test cases passed")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/Desktop/akhils ai coding/7.5.py"
All test cases passed
PS C:\Users\achyu\OneDrive\Desktop\akhils ai 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())
```

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 the Visual Studio Code interface. The left sidebar has icons for file operations, search, and other extensions. The main area shows a Python script named '7.5.py'. The code contains several assert statements and a print statement. The terminal at the bottom shows the command being run and the output 'All test cases passed'.

```

81     #     return "Numbers: " + str([1, 2, 3])
82
83 # assert combine() == "Numbers: [1, 2, 3]"
84 # assert "A" + str([1]) == "A[1]"
85 # assert "List: " + str([]) == "List: []"
86
87 # print("All test cases passed")
88
89 def repeat_text():
90     return "Hello" * int(2.5)
91
92 assert repeat_text() == "HelloHello"
93 assert "A" * int(3.9) == "AAA"
94 assert "Hi" * int(1.1) == "Hi"
95
96 print("All test cases passed")
97
98
99
100
101
102

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/7.5.py"
All test cases passed
PS C:\Users\achyu\OneDrive\Desktop\akhils ai 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())
```

Requirements:

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

A screenshot of a code editor window titled "7.5.py". The code contains several assert statements and a compute function. The terminal tab shows the output of running the script, which includes the message "All test cases passed".

```
91 # assert repeat_text() == "HelloHello"
92 # assert "A" * int(3.9) == "AAA"
93 # assert "Hi" * int(1.1) == "Hi"
94
95
96 # print("All test cases passed")
97
98 def compute():
99     value = 0
100    return value + 10
101
102 assert compute() == 10
103 assert 0 + 5 == 5
104 assert (0 + 0) == 0
105
106 print("All test cases passed")
107
108
109
110
111
112
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:\ive\Desktop\akhils ai coding\7.5.py"
All test cases passed
PS C:\Users\achyu\OneDrive\Desktop\akhils ai 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())
```

Requirements:

- Explain why input is always string.
- Fix using int() conversion.
- Verify with assert test cases.

```
File Edit Selection View Go ... ⏪ ⏩ 🔍 akhils ai coding
7.5.py ✎
7.5.py > ...
101
102     # assert compute() == 10
103     # assert 0 + 5 == 5
104     # assert (0 + 0) == 0
105
106     # print("All test cases passed")
107
108
109 def sum_two_numbers(a, b):
110     return int(a) + int(b)
111
112 assert sum_two_numbers("2", "3") == 5
113 assert sum_two_numbers("10", "20") == 30
114 assert sum_two_numbers("0", "5") == 5
115
116 print("All test cases passed")
117
118
119
120
121
122
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/7.5.py"
All test cases passed
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```