

## AI ASSISTED CODING

### LAB-7.5

Katta Lasya

2303A51724

Batch-11

#### Task 1 (Mutable Default Argument – Function Bug)

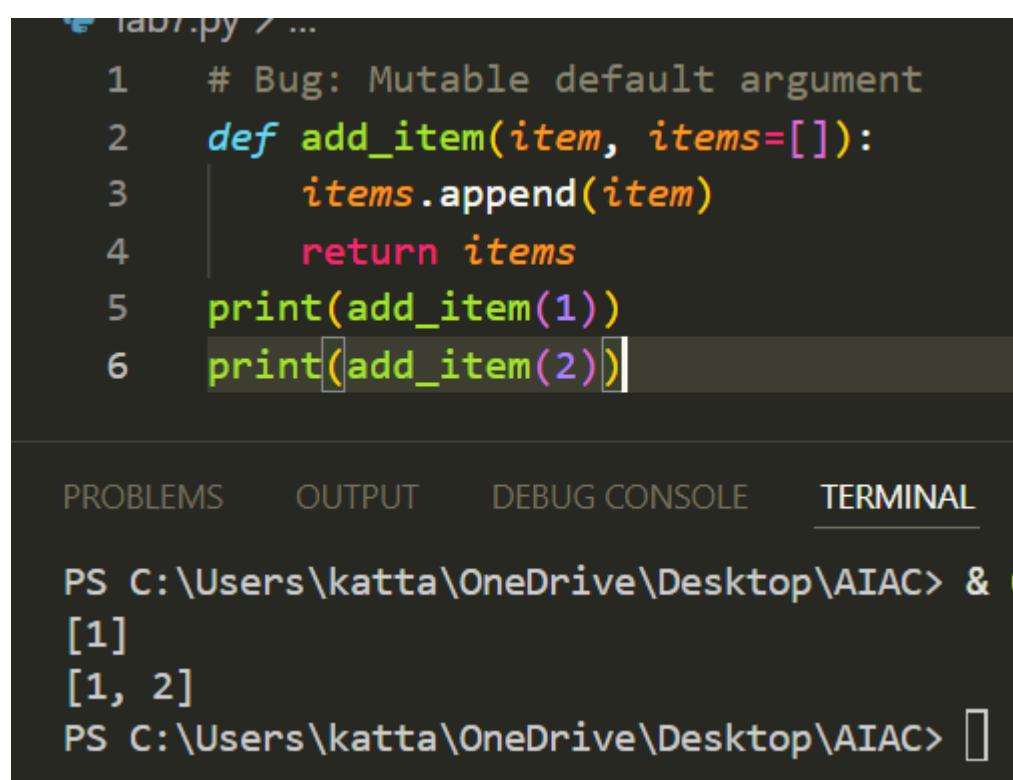
Task: Analyze given code where a mutable default argument causes unexpected behavior. Use AI to fix it.

```
# 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.

#### Given Code and Output:



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

```
lab7.py > ...
1 # Bug: Mutable default argument
2 def add_item(item, items=[]):
3     items.append(item)
4     return items
5 print(add_item(1))
6 print(add_item(2))
```

Below the code editor, there is a terminal window showing the output of running the script:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\katta\OneDrive\Desktop\AIAC> & lab7.py
[1]
[1, 2]
PS C:\Users\katta\OneDrive\Desktop\AIAC> []
```

### 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

Given Code and Output:

```
lab7.py > check_sum

1 # Bug: Floating point precision issue
2 import math
3
4 def check_sum():
5     return math.isclose(0.1 + 0.2, 0.3)
6 print(check_sum())

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\katta\OneDrive\Desktop\AIAC> & C:/User
True
PS C:\Users\katta\OneDrive\Desktop\AIAC> 
```

### 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.

Given Code and Output:

The screenshot shows a code editor with a dark theme. In the top-left corner, there's a file icon followed by 'lab7.py > 📁 countdown'. Below this is a code block with the following Python code:

```
1 def countdown(n):
2     if n == 0:
3         return
4     print(n)
5     countdown(n-1)
6 countdown(5)
```

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

```
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C:\>
5
4
3
2
1
PS C:\Users\katta\OneDrive\Desktop\AIAC> □
```

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.

Given Code and output:

```
.lab7.py > get_value
1 def get_value():
2     data = {"a": 1, "b": 2}
3     return data.get("c", "Key not found")
4 print(get_value())
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C:/Users,
Key not found
PS C:\Users\katta\OneDrive\Desktop\AIAC> 
```

#### 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.

Given Code and Output:

```
.lab7.py > ...
1  def loop_example():
2      i = 0
3      while i < 5:
4          print(i)
5          i += 1
6  loop_example()

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C:
0
1
2
3
4
PS C:\Users\katta\OneDrive\Desktop\AIAC> 
```

#### 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.

Given Code:

```
lab7.py > ...
1  a, b, _ = (1, 2, 3)
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

**Expected Output : Consistent indentation applied.**

**Given Code and Output:**

```
.lab7.py > ...
1 def func():
2     x = 5
3     y = 10
4     return x+y
5 print(func())
PROBLEMS OUTPUT DEBUG CONSOLE
PS C:\Users\katta\OneDrive\Desktop> lab7.py
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**

**Given Code and Output:**

```
.lab7.py
1 import math
2 print(math.sqrt(16))
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C
4.0
PS C:\Users\katta\OneDrive\Desktop\AIAC>
```