

# **LAB ASSIGNMENT-7.5**

NAME:M.ABHISHEK

2303A51762

BATCH-11

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

Lab Objectives:

- To identify and correct syntax, logic, and runtime errors in

Week4 -Monday:

Python programs using AI tools.

- To understand common programming bugs and AI-assisted debugging suggestions.
- To evaluate how AI explains, detects, and fixes different types of coding errors.
- To build confidence in using AI to perform structured debugging practices.

Lab Outcomes (LOs):After completing this lab, students will be able to:

- Use AI tools to detect and correct syntax, logic, and runtime errors.
- Interpret AI-suggested bug fixes and explanations.
- Apply systematic debugging strategies supported by AI-generated insights.

Refactor buggy code using responsible and reliable programming patterns.

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

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

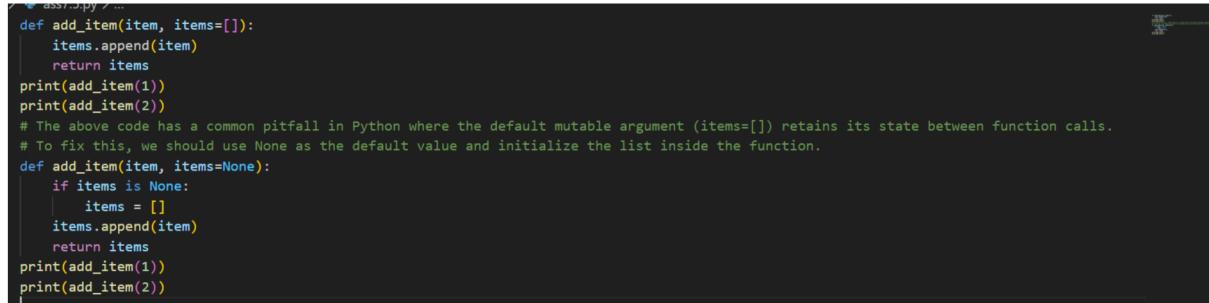
```
# Bug: Mutable default argument
```

```
def add_item(item, items=[]):
```

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

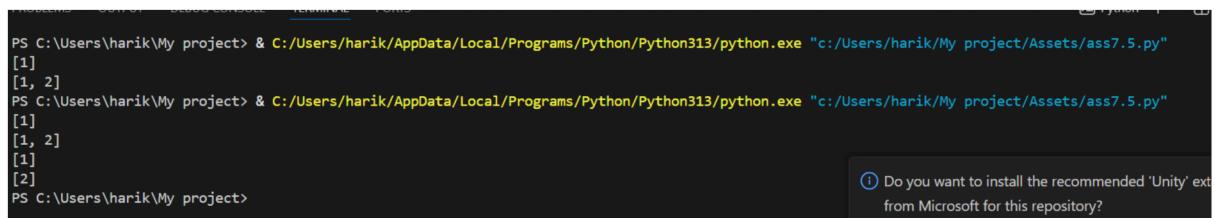
```
return items  
print(add_item(1))  
print(add_item(2))
```

Expected Output: Corrected function avoids shared list bug.



```
* ass7.5.py > ...  
def add_item(item, items=[]):  
    items.append(item)  
    return items  
print(add_item(1))  
print(add_item(2))  
# The above code has a common pitfall in Python where the default mutable argument (items=[]) retains its state between function calls.  
# To fix this, we should use None as the default value and initialize the list inside the function.  
def add_item(item, items=None):  
    if items is None:  
        items = []  
    items.append(item)  
    return items  
print(add_item(1))  
print(add_item(2))
```

Output:



```
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/ass7.5.py"  
[1]  
[1, 2]  
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/ass7.5.py"  
[1]  
[1, 2]  
[1]  
[2]  
PS C:\Users\harik\My project>
```

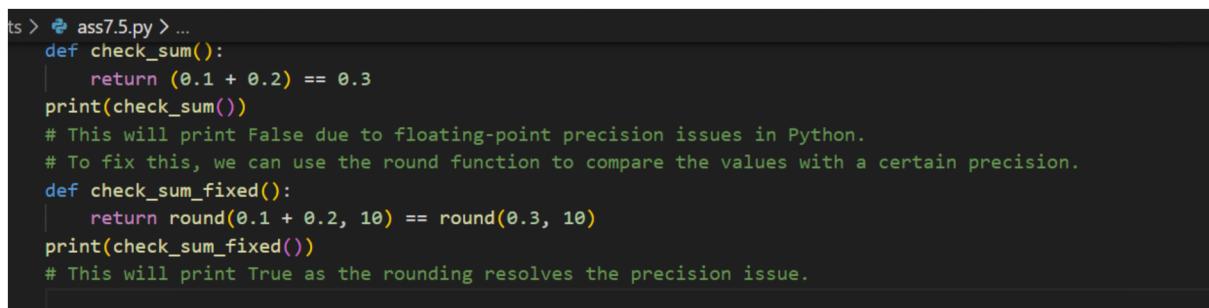
## Task 2 (Floating-Point Precision Error)

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



```
ts > ass7.5.py > ...  
def check_sum():  
    return (0.1 + 0.2) == 0.3  
print(check_sum())  
# This will print False due to floating-point precision issues in Python.  
# To fix this, we can use the round function to compare the values with a certain precision.  
def check_sum_fixed():  
    return round(0.1 + 0.2, 10) == round(0.3, 10)  
print(check_sum_fixed())  
# This will print True as the rounding resolves the precision issue.
```

Output:

```
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/ass7.5.py"
False
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/ass7.5.py"
False
True
PS C:\Users\harik\My project>
```

### 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

```
#Fixed Code:
def countdown(n):
    if n <= 0:
        print("Countdown finished!")
        return
    print(n)
    return countdown(n-1)
```

Output:

```
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/ass7.5.py"
5
4
3
2
1
Countdown finished!
PS C:\Users\harik\My project>
```

Do you want to install the recommended 'Unity' extension from Microsoft for this repository?

Install Show Recommendations

### 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.

```
# correct the above code with .get() or error handling.
def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c", 0)
print(get_value())
```

Output:

```
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/ass7.5.py"
0
0
0
0
0
0
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/ass7.5.py"
0
0
0
0
0
0
PS C:\Users\harik\My project>
```

Do you want to install the recommended 'Unity' extension from Microsoft for this repository?

### 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.

```

def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
#correct the above code and handle the errors
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
loop_example()
loop_example()
#Corrected loop increments i.
def loop_example():
    i = 0

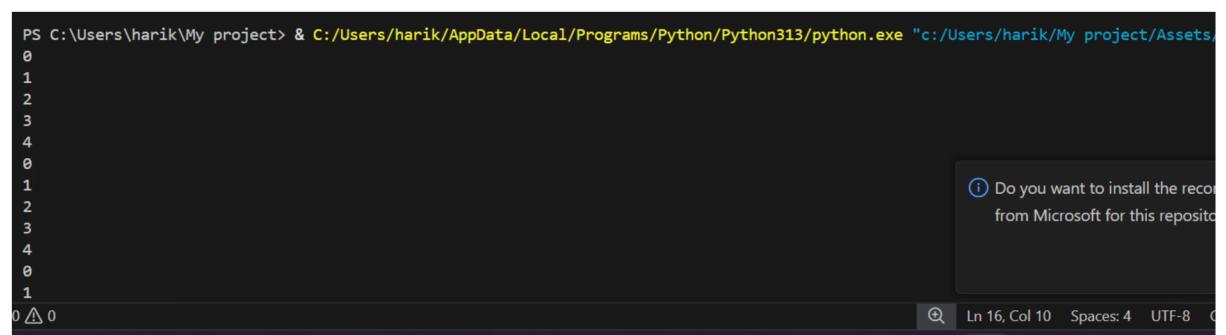
```

```

#Corrected loop increments i.
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
loop_example()
loop_example()

```

Output:



```

PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/
0
1
2
3
4
0
1
2
3
4
0
1
0 ▲ 0

```

Do you want to install the recommended updates from Microsoft for this repository?

Ln 16, Col 10 Spaces: 4 UTF-8

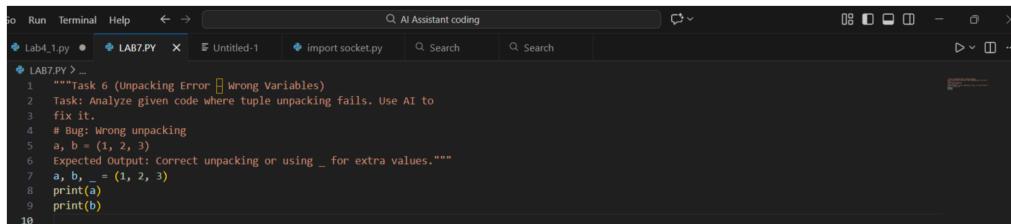
## Task 6 (Unpacking Error – Wrong Variables)

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

```
# Bug: Wrong unpacking
```

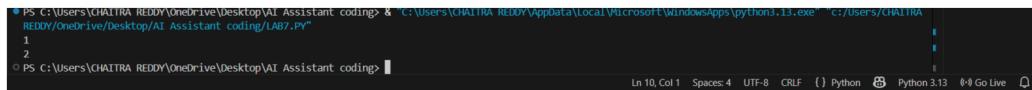
```
a, b = (1, 2, 3)
```

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



```
Lab4_1.py  LAB7.PY  Untitled-1  import socket.py  Search  Search
LAB7.PY > ...
1  """Task 6 (Unpacking Error) Wrong Variables)
2  Task: Analyze given code where tuple unpacking fails. Use AI to
3  fix it.
4  # Bug: Wrong unpacking
5  a, b = (1, 2, 3)
6  Expected Output: Correct unpacking or using _ for extra values."""
7  a, b, _ = (1, 2, 3)
8  print(a)
9  print(b)
10
```

Output:



```
PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> & "C:\Users\CHAITRA REDDY\AppData\Local\Microsoft\WindowsApps\python3.13.exe" "c:/Users/CHAITRA REDDY/OneDrive/Desktop/AI Assistant coding/LAB7.PY"
1
2
3
4
5
6
7
8
9
10
```

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

# Bug: Mixed indentation

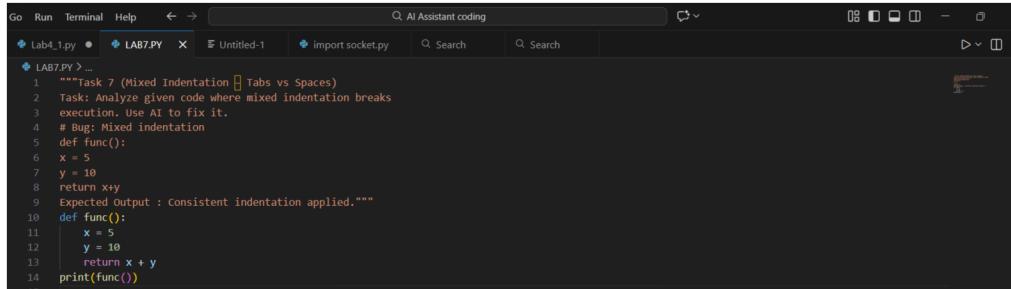
```
def func():
```

```
    x = 5
```

```
    y = 10
```

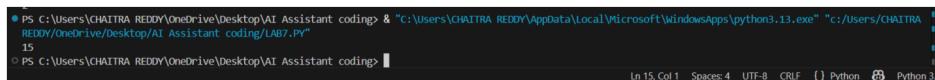
```
    return x+y
```

Expected Output : Consistent indentation applied.



```
Lab4_1.py  LAB7.PY  Untitled-1  import socket.py  Search  Search
LAB7.PY > ...
1  """Task 7 (Mixed Indentation) Tabs vs Spaces)
2  Task: Analyze given code where mixed indentation breaks
3  execution. Use AI to fix it.
4  # Bug: Mixed indentation
5  def func():
6      x = 5
7      y = 10
8      return x+y
9  Expected Output : Consistent indentation applied."""
10 def func():
11     x = 5
12     y = 10
13     return x + y
14 print(func())
15
```

Output:



```
PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> & "C:\Users\CHAITRA REDDY\AppData\Local\Microsoft\WindowsApps\python3.13.exe" "c:/Users/CHAITRA REDDY/OneDrive/Desktop/AI Assistant coding/LAB7.PY"
15
16
17
18
19
20
```

**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

The screenshot shows a code editor window titled "AI Assistant coding". It has tabs for "Lab4\_1.py", "LAB7.PY" (which is the active tab), and "Untitled-1". Below the tabs is a search bar. The code in the "LAB7.PY" tab is as follows:

```
1  """Task 8 (Import Error) Wrong Module Usage
2  Task: Analyze given code with incorrect import. Use AI to fix.
3  # Bug: Wrong import
4  import maths
5  print(maths.sqrt(16))
6  Expected Output: Corrected to import math"""
7  import math
8  print(math.sqrt(16))
9
```

Output:

The screenshot shows a terminal window with two entries. The first entry is a blue-highlighted command to run the file "LAB7.PY" using Python 3.13. The second entry is a red-highlighted command to run the same file again. The status bar at the bottom indicates the file is in line 9, column 1, with 4 spaces, and the encoding is UTF-8.

```
PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> & "C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding\LAB7.PY"
4.0
PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding>
```