

# Lab – 7.5

## Course : AI Assisted Coding

*Topic:- Error Debugging with AI: Systematic approaches to finding and fixing bugs*

---

Student Details:

Name : Mohammed Sabir

Hall Ticket No : 2303A51506

Batch 22

---

### Task-01

Final Optimal Prompt

---

Fix only the errors in the voting eligibility code and add validation for invalid or non-numeric age input without changing the main logic

Code Screenshot

---



```
1 #---Lab-7-Task-01/05--- Chat Ctrl+L Edit Ctrl+I
fix the
Add Context...
2
# Bug: Mutable default argument
def add_item(item, items=[]):
    items.append(item)
    return items
3 # Bug: Mutable default argument - FIXED
4 def add_item(item, items=None):
5     if items is None:
6         items = []
7 print(add_item(1))
8 print(add_item(2))
9
```

Output Screenshot

---

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\6th Sem 2025-26\AI-Assisted Coding> & "D:\pytho
26/AI-Assisted Coding/Assignment-07(Lab-7.5).py"
[1]
[2]
PS D:\6th Sem 2025-26\AI-Assisted Coding> █
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

---

The bug occurs because the default list is shared across all function calls. AI helps identify the issue and fixes it by replacing the mutable default with None. A new list is created on each call, preventing unexpected behaviour.

## Task-02

Final Optimal Prompt

---

Correct the string vowel/consonant code and add error handling for numbers, symbols, or empty input while keeping the original structure.

Code Screenshot

---

```
# Bug: Floating point precision issue
def check_sum():
    return (0.1 + 0.2) == 0.3
... return abs((0.1 + 0.2) - 0.3) < 1e-9
print(check_sum())
```

Output Screenshot

---

```
PS D:\6th Sem 2025-26\AI-Assisted
26/AI-Assisted Coding/Assignment
True
PS D:\6th Sem 2025-26\AI-Assisted
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

---

Floating-point arithmetic cannot reliably compare decimal values directly. AI suggests using a tolerance check (`abs(diff) < epsilon`) to avoid precision errors. This ensures a correct and reliable comparison.

## Task-03

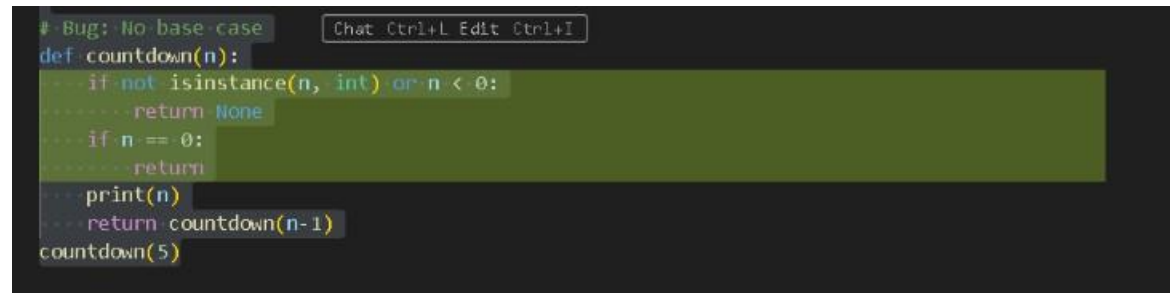
### Final Optimal Prompt

---

Fix errors in the class-based library program and add simple input validation without rewriting or optimizing the logic

### Code Screenshot

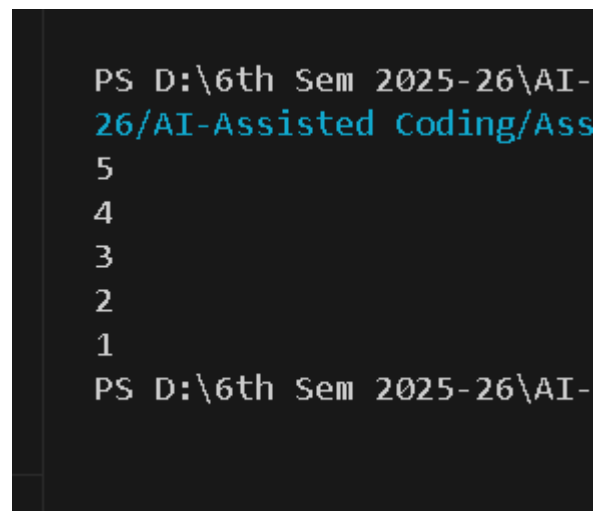
---

A screenshot of a code editor showing a Python function named 'countdown'. The function has a docstring that says '# Bug: No base case'. The function definition is: 'def countdown(n):'. Inside the function, there is an 'if' statement: 'if not isinstance(n, int) or n < 0:'. This is followed by an indented 'return None' statement. Then, there is another 'if' statement: 'if n == 0:'. This is followed by an indented 'return' statement. After the 'if' statements, there is an indented 'print(n)' statement and an indented 'return countdown(n-1)' statement. Finally, the function is called with 'countdown(5)'.

```
# Bug: No base case
def countdown(n):
    if not isinstance(n, int) or n < 0:
        return None
    if n == 0:
        return
    print(n)
    return countdown(n-1)
countdown(5)
```

### Output Screenshot

---

A screenshot of a terminal window showing the output of the 'countdown' function. The output is a sequence of numbers: 5, 4, 3, 2, 1, followed by a new line and the prompt 'PS D:\6th Sem 2025-26\AI-'.

```
PS D:\6th Sem 2025-26\AI-
5
4
3
2
1
PS D:\6th Sem 2025-26\AI-
```

### Explanation/Justification/Observation (100 words / 5 – 6 sentence)

---

The original function lacked a base case, causing infinite recursion. AI introduces a stopping condition ( $n \leq 0$ ), allowing the function to terminate safely.

## Task-04

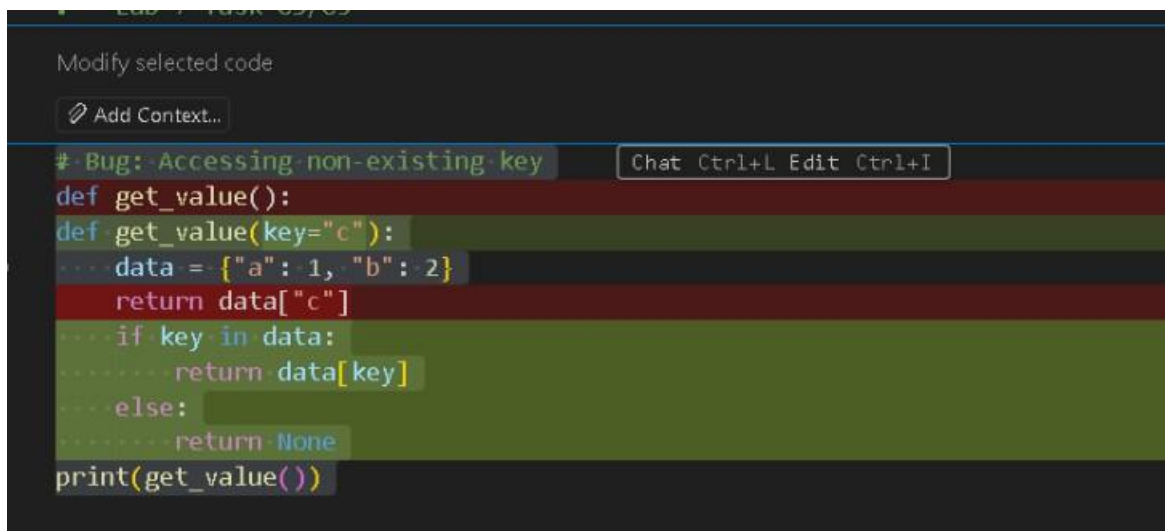
### Final Optimal Prompt

---

Correct the attendance class code and add basic validation for wrong or empty inputs without altering the program design

### Code Screenshot

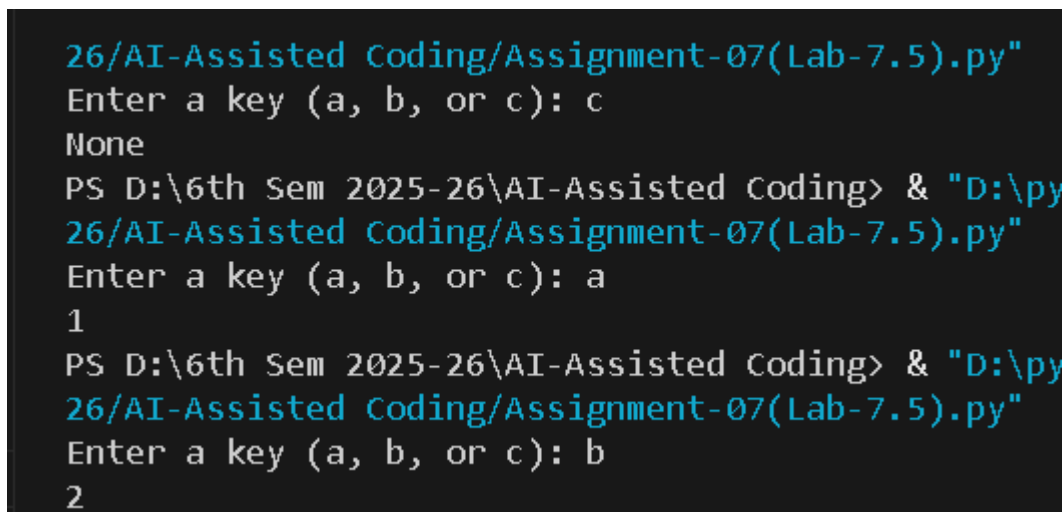
---



```
# Bug: Accessing non-existing key
def get_value():
def get_value(key="c"):
    data = {"a": 1, "b": 2}
    return data.get(key)
    if key in data:
        return data[key]
    else:
        return None
print(get_value())
```

### Output Screenshot

---



```
26/AI-Assisted Coding/Assignment-07(Lab-7.5).py"
Enter a key (a, b, or c): c
None
PS D:\6th Sem 2025-26\AI-Assisted Coding> & "D:\py
26/AI-Assisted Coding/Assignment-07(Lab-7.5).py"
Enter a key (a, b, or c): a
1
PS D:\6th Sem 2025-26\AI-Assisted Coding> & "D:\py
26/AI-Assisted Coding/Assignment-07(Lab-7.5).py"
Enter a key (a, b, or c): b
2
```

### Explanation/Justification/Observation (100 words / 5 – 6 sentence)

---

Accessing a missing key raises a Key Error. AI fixes this using .get() with a default message, preventing crashes and improving program reliability.

## Tack-05

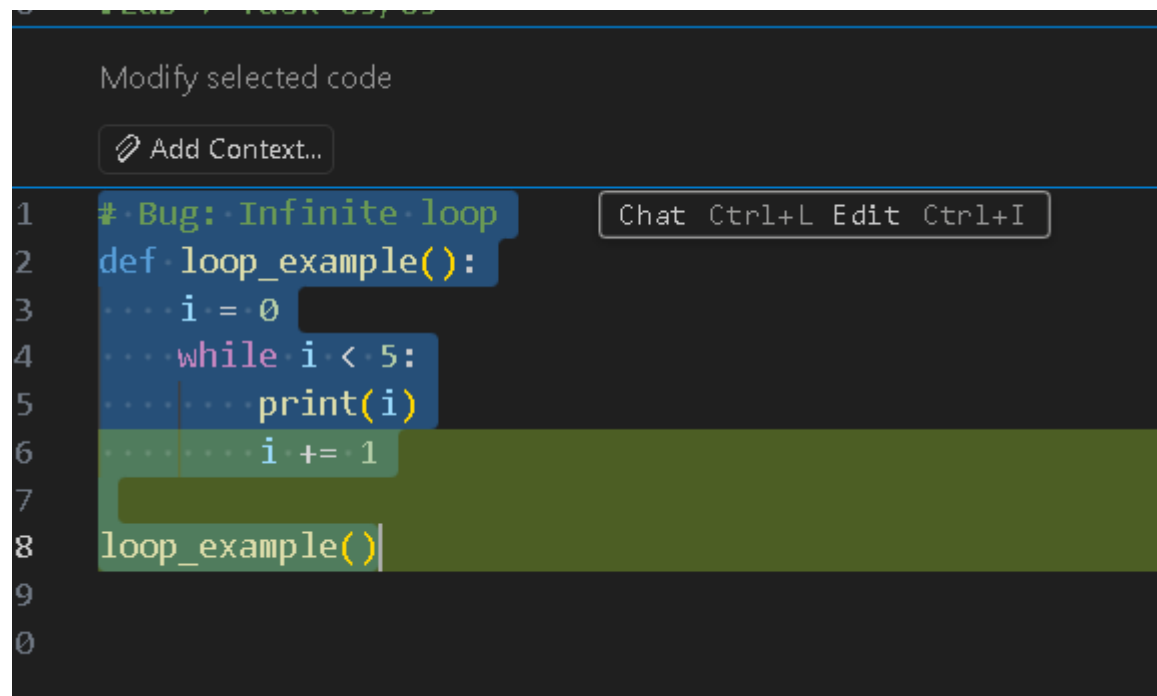
### Final Optimal Prompt

---

Fix the ATM menu errors and add checks for invalid menu choices or non-numeric inputs while keeping the same program flow

### Code Screenshot

---



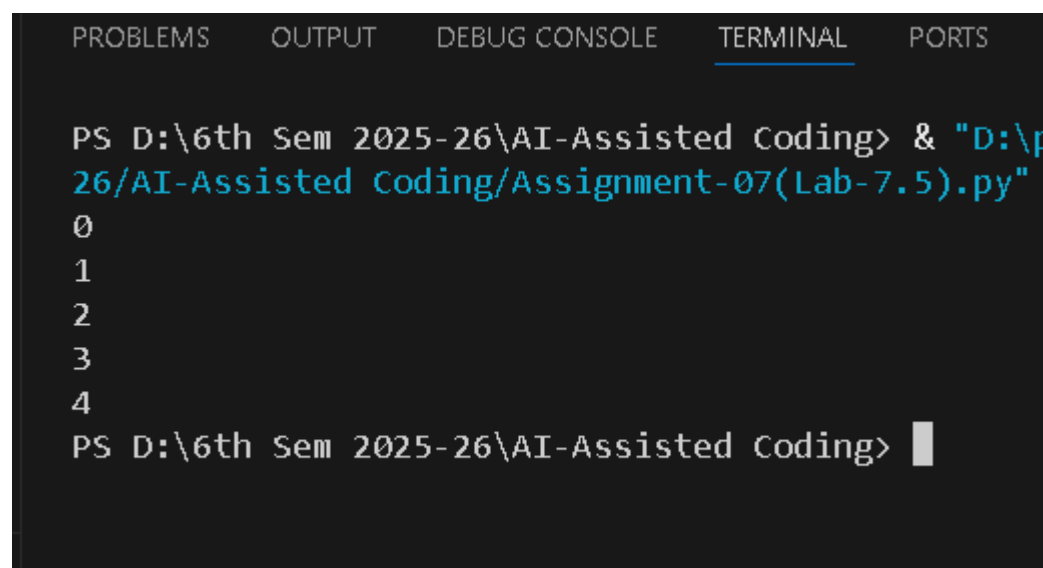
The screenshot shows a code editor with a dark theme. At the top, there's a search bar with the text "Modify selected code" and a button "Add Context...". Below this, a code snippet is displayed with line numbers 1 through 9. The code is as follows:

```
1 # Bug: Infinite loop
2 def loop_example():
3     i = 0
4     while i < 5:
5         print(i)
6         i += 1
7
8 loop_example()
9
10
```

The code is highlighted with a blue selection bar. A tooltip "Chat Ctrl+L Edit Ctrl+I" is visible next to the code.

### Output Screenshot

---



The screenshot shows a terminal window with the following tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The terminal output is as follows:

```
PS D:\6th Sem 2025-26\AI-Assisted Coding> & "D:\p
26\AI-Assisted Coding\Assignment-07(Lab-7.5).py"
0
1
2
3
4
PS D:\6th Sem 2025-26\AI-Assisted Coding>
```

### Explanation/Justification/Observation (100 words / 5 – 6 sentence)

---

The loop condition was correct, but the variable never changed, causing an infinite loop. AI fixes it by incrementing `i` inside the loop.

## Tack-06

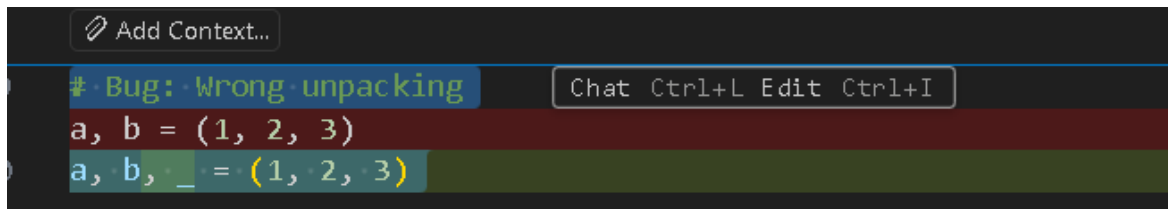
Final Optimal Prompt

---

Correct the tuple unpacking issue without altering the original values

Code Screenshot

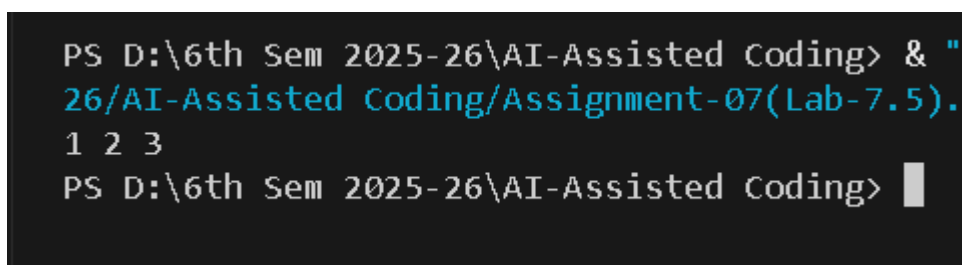
---



```
# Bug: Wrong unpacking
a, b = (1, 2, 3)
a, b, _ = (1, 2, 3)
```

Output Screenshot

---



```
PS D:\6th Sem 2025-26\AI-Assisted Coding> & "
26/AI-Assisted Coding/Assignment-07(Lab-7.5).
1 2 3
PS D:\6th Sem 2025-26\AI-Assisted Coding> █
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

---

The bug occurs because the tuple contains 3 values but only 2 variables. AI corrects the mismatch by adding another variable or using \_ to ignore the extra value.

## Tack-07

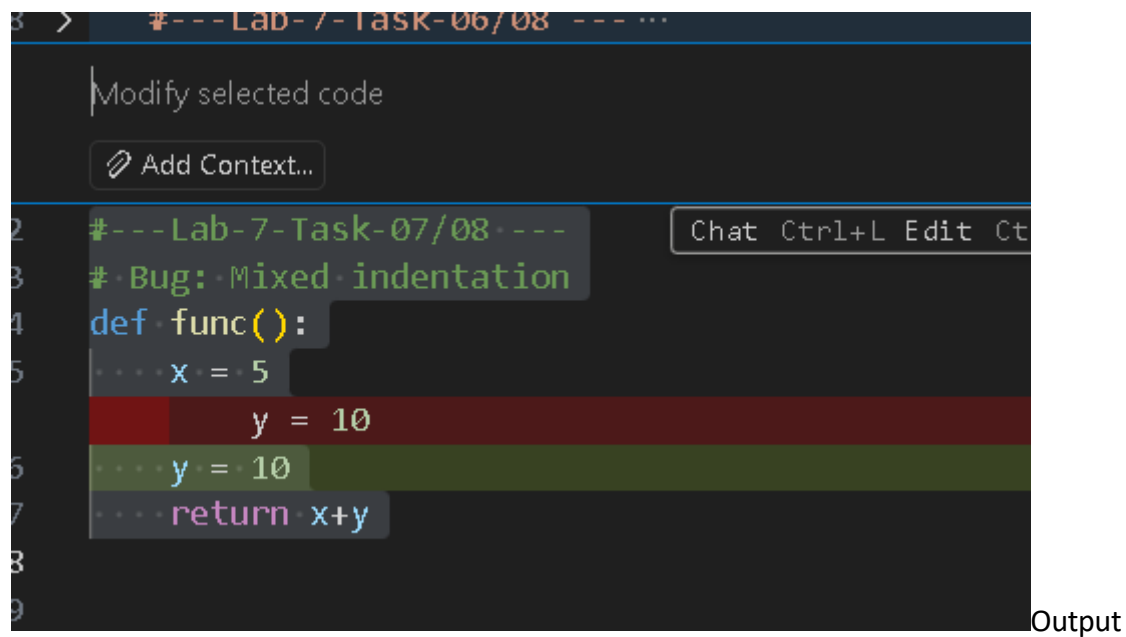
Final Optimal Prompt

---

Fix the mixed indentation so the code runs properly; do not change any logic

Code Screenshot

---



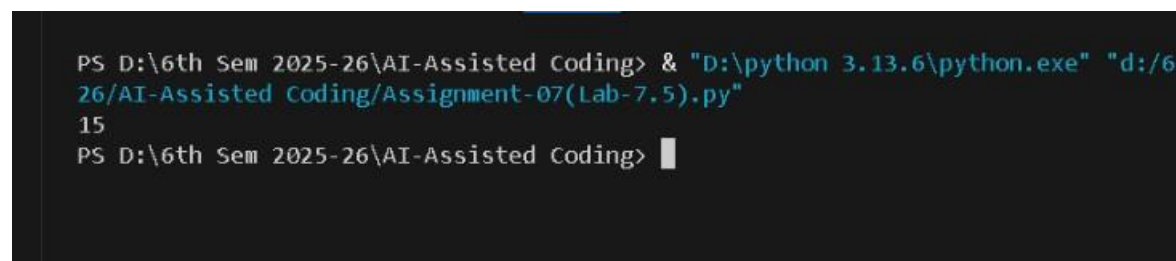
```
#---Lab-7-Task-07/08---
# Bug: Mixed indentation
def func():
    ... x = 5
    y = 10
    ... y = 10
    ... return x+y
```

Output

```
15
```

Screenshot

---



```
PS D:\6th Sem 2025-26\AI-Assisted Coding> & "D:\python 3.13.6\python.exe" "d:/626/AI-Assisted Coding/Assignment-07(Lab-7.5).py"
15
PS D:\6th Sem 2025-26\AI-Assisted Coding>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

---

Tabs and spaces were mixed, causing an indentation error. AI normalizes indentation using consistent spaces, ensuring the code runs correctly.

## Tack-08

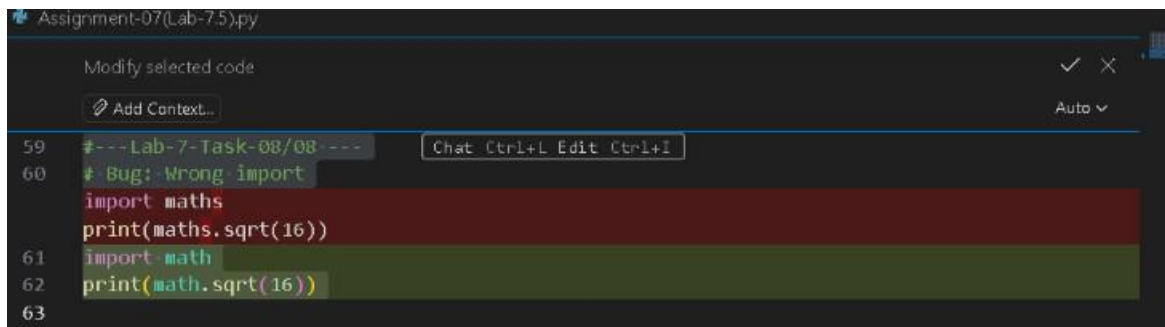
Final Optimal Prompt

---

Fix only the module import error so the sqrt function works correctly

Code Screenshot

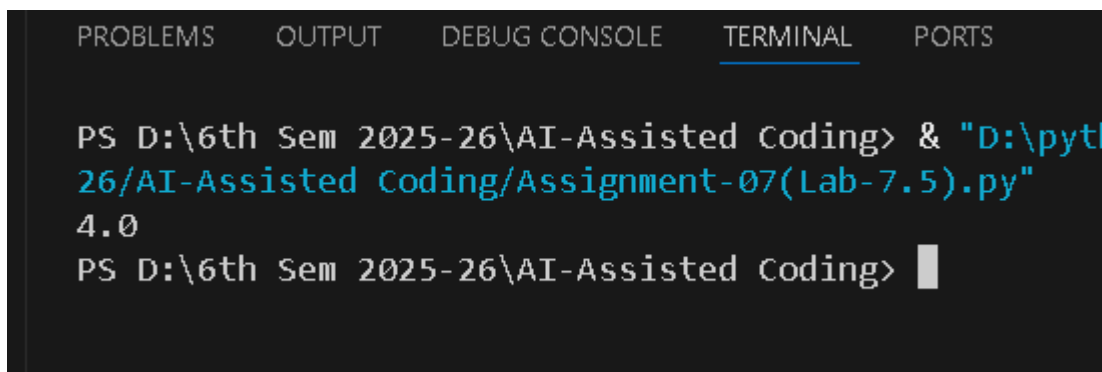
---



```
Assignment-07(Lab-7.5).py
Modify selected code
Add Context...
Auto v
59 #---Lab-7-Task-08/08---
60 # Bug: Wrong import
import maths
print(maths.sqrt(16))
61 import math
62 print(math.sqrt(16))
63
```

Output Screenshot

---



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\6th Sem 2025-26\AI-Assisted Coding> & "D:\python\26\AI-Assisted Coding\Assignment-07(Lab-7.5).py"
4.0
PS D:\6th Sem 2025-26\AI-Assisted Coding>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

---

The module name was incorrect (maths does not exist). AI corrects it to math, ensuring proper module import and execution.