

AI ASSISTANT CODING

ASSIGNMENT-7

Name: Kommu Madhupriya

Hallticket:2303A51583

Batch:22

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 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)
```

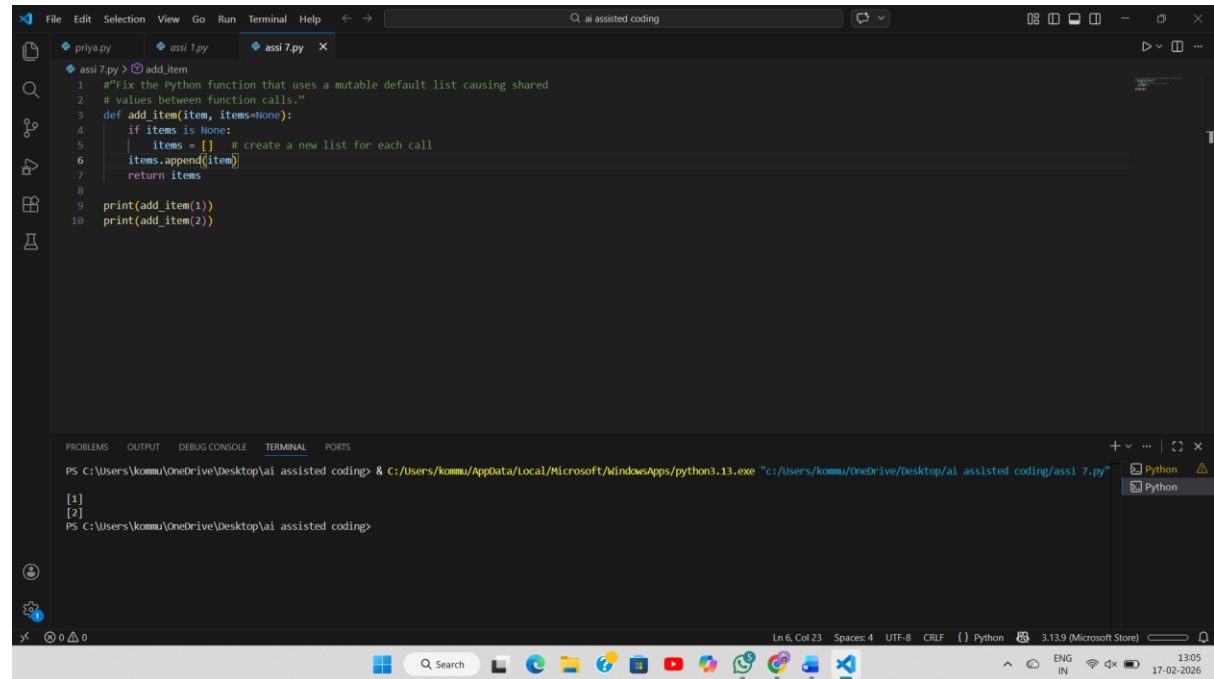
return items

print(add_item(1))

print(add_item(2))

Expected Output: Corrected function avoids shared list bug.

code:



The screenshot shows a Visual Studio Code editor window with a Python file named `assi 7.py`. The code defines a function `add_item` that takes an `item` and an optional `items` list. If `items` is `None`, it creates a new list. It then appends the `item` to the list and returns it. The script calls `print(add_item(1))` and `print(add_item(2))`. The terminal at the bottom shows the command to run the script, and the output displays `[1]` and `[2]` on separate lines, indicating that each function call returns a new list instead of sharing a common one.

```
1 """Fix the Python function that uses a mutable default list causing shared
2 # values between function calls."""
3 def add_item(item, items=None):
4     if items is None:
5         items = [] # create a new list for each call
6         items.append(item)
7     return items
8
9 print(add_item(1))
10 print(add_item(2))
```

PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & C:\Users\kommu\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/Users/kommu/OneDrive/Desktop/ai assisted coding/assi 7.py"

[1]
[2]
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>

Task 2 (Floating-Point Precision Error)

Task: Analyze given code where floating-point comparison fails.

Use AI to correct with tolerance.

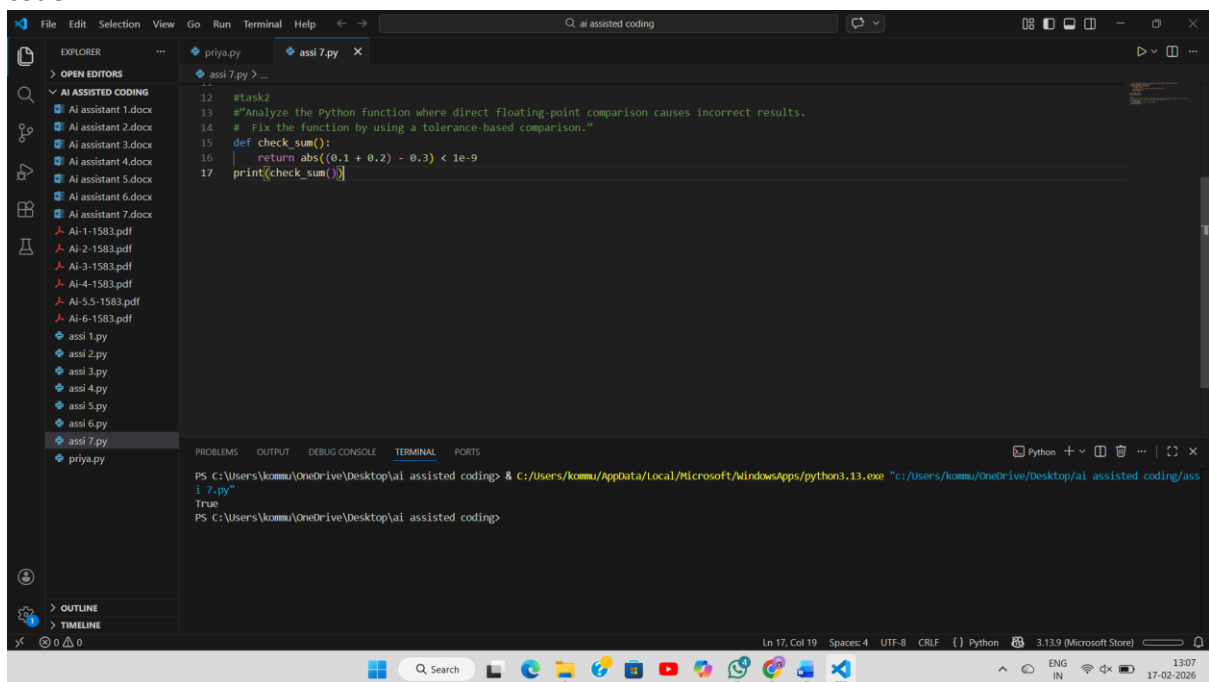
Bug: Floating point precision issue

```
def check_sum():
```

```
    return (0.1 + 0.2) == 0.3
```

```
print(check_sum())
```

Expected Output: Corrected function code:



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

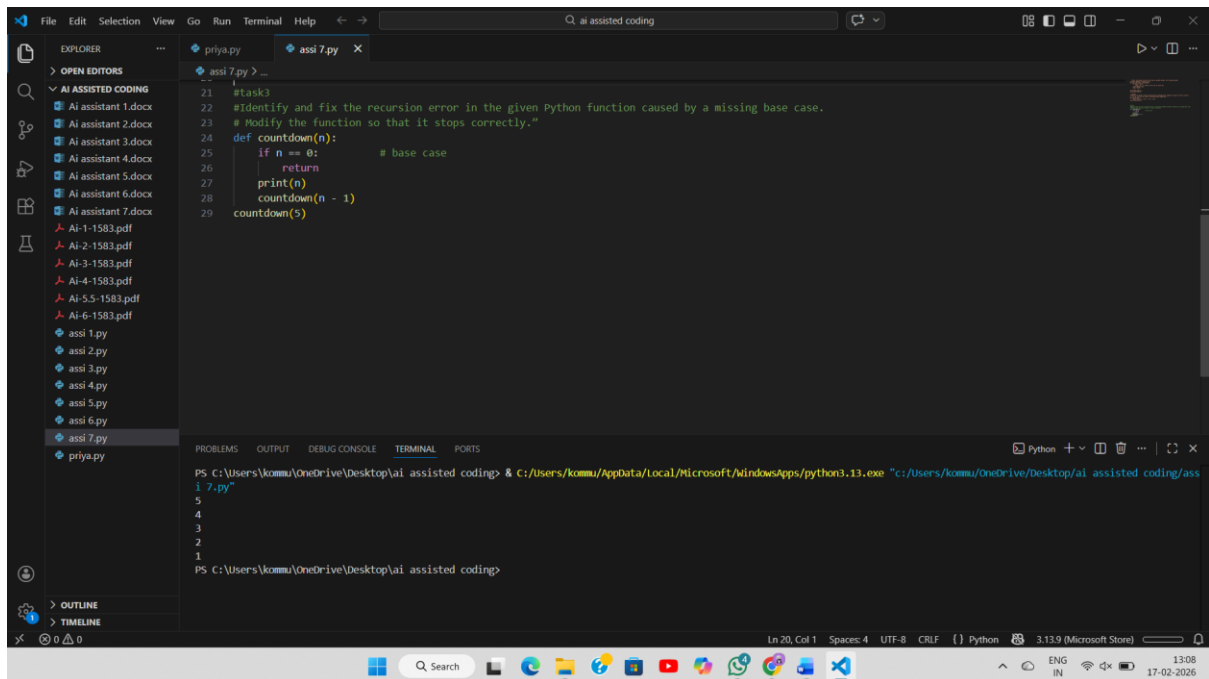
```
def countdown(n):
```

```
    print(n)
```

```
    return countdown(n-1)
```

```
countdown(5)
```

Expected Output : Correct recursion with stopping condition.
code:



```
21 #task3
22 #Identify and fix the recursion error in the given Python function caused by a missing base case.
23 # Modify the function so that it stops correctly."
24 def countdown(n):
25     if n == 0: # base case
26         return
27     print(n)
28     countdown(n - 1)
29     countdown(5)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & C:/Users/kommuk/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/kommuk/OneDrive/Desktop/ai assisted coding/ass
7.py"
5
4
3
2
1
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

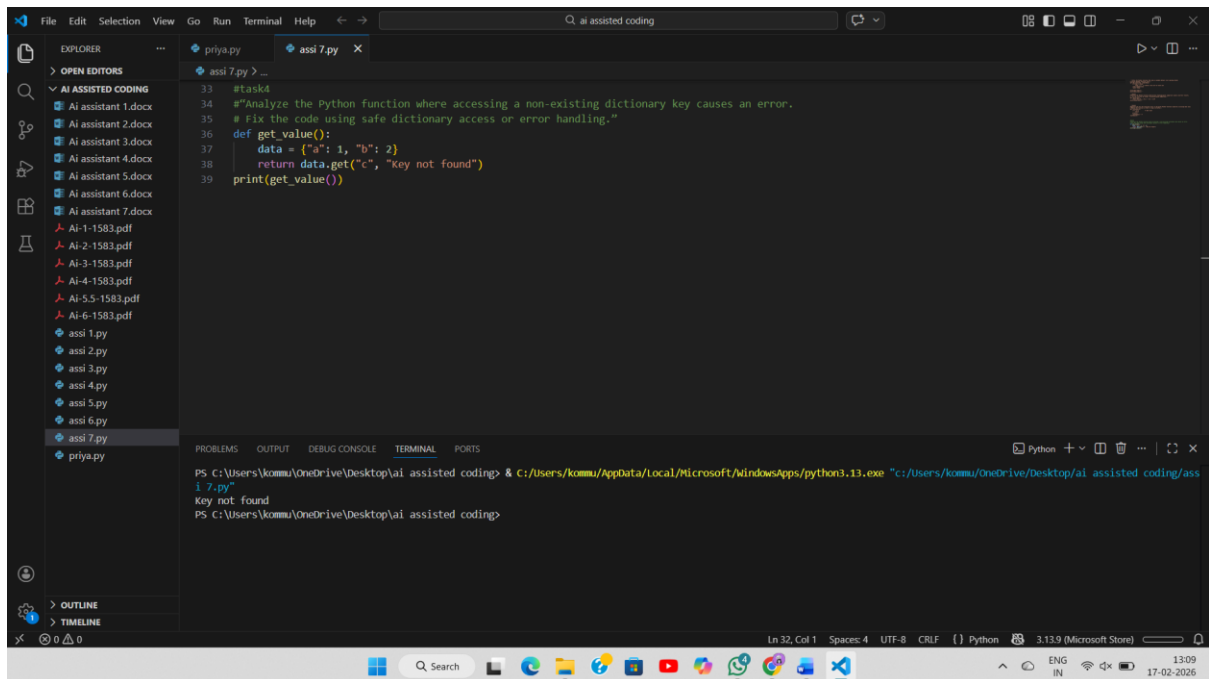
Task 4 (Dictionary Key Error)

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.
code:



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():
```

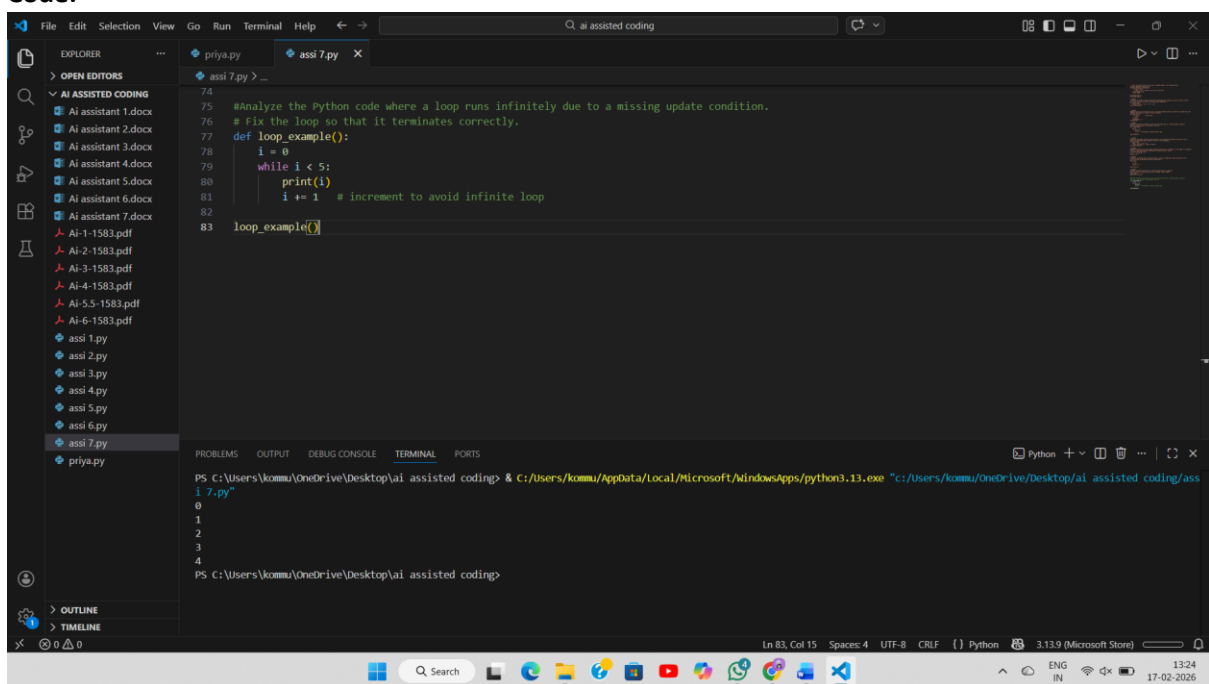
```
    i = 0
```

```
    while i < 5:
```

```
        print(i)
```

Expected Output: Corrected loop increments i.

Code:



```
File Edit Selection View Go Run Terminal Help  
ai assisted coding  
EXPLORER  
AI ASSISTED CODING  
Ai assistant 1.docx  
Ai assistant 2.docx  
Ai assistant 3.docx  
Ai assistant 4.docx  
Ai assistant 5.docx  
Ai assistant 6.docx  
Ai assistant 7.docx  
Ai-1-1583.pdf  
Ai-2-1583.pdf  
Ai-3-1583.pdf  
Ai-4-1583.pdf  
Ai-5-1583.pdf  
Ai-6-1583.pdf  
assi 1.py  
assi 2.py  
assi 3.py  
assi 4.py  
assi 5.py  
assi 6.py  
assi 7.py  
priya.py  
OUTLINE  
TIMELINE  
0 0 0  
74  
75 #Analyze the Python code where a loop runs infinitely due to a missing update condition.  
76 # Fix the loop so that it terminates correctly.  
77 def loop_example():  
78     i = 0  
79     while i < 5:  
80         print(i)  
81         i += 1 # increment to avoid infinite loop  
82  
83 loop_example()  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
Python  
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & c:/Users/komm/OneDrive/Desktop/ai assisted coding/ass  
i 7.py"  
0  
1  
2  
3  
4  
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>  
Ln 83, Col 15 Spaces: 4 UTF-8 CRLF Python 3.13.9 (Microsoft Store) 13:24 17-02-2026
```

Task 6 (Unpacking Error – Wrong Variables)

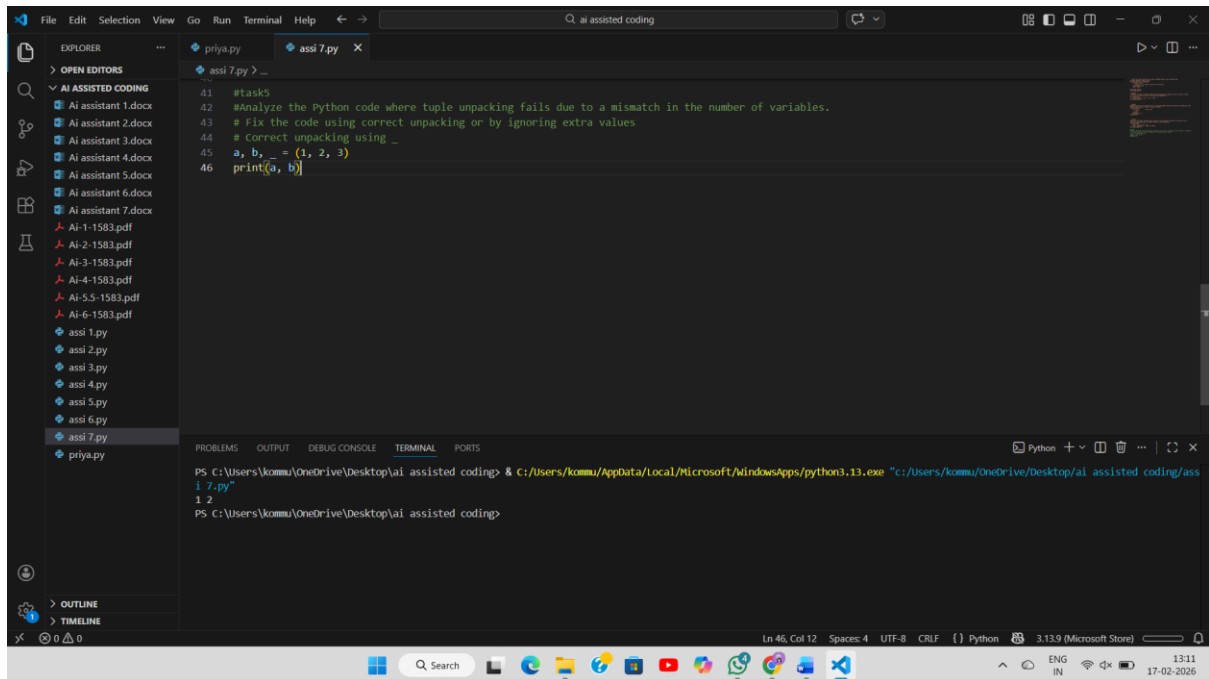
Task: Analyze given code where tuple unpacking fails. Use AI to fix it.

Bug: Wrong unpacking

a, b = (1, 2, 3)

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

Code:



```
41 #task5
42 #Analyze the Python code where tuple unpacking fails due to a mismatch in the number of variables.
43 # Fix the code using correct unpacking or by ignoring extra values
44 # Correct unpacking using _
45 a, b, _ = (1, 2, 3)
46 print(a, b)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + Python 3.13.9 (Microsoft Store)

PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & c:/Users/komm/OneDrive/Desktop/ai assisted coding/assi 7.py

1 2

PS C:\Users\kommu\OneDrive\Desktop\ai assisted 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

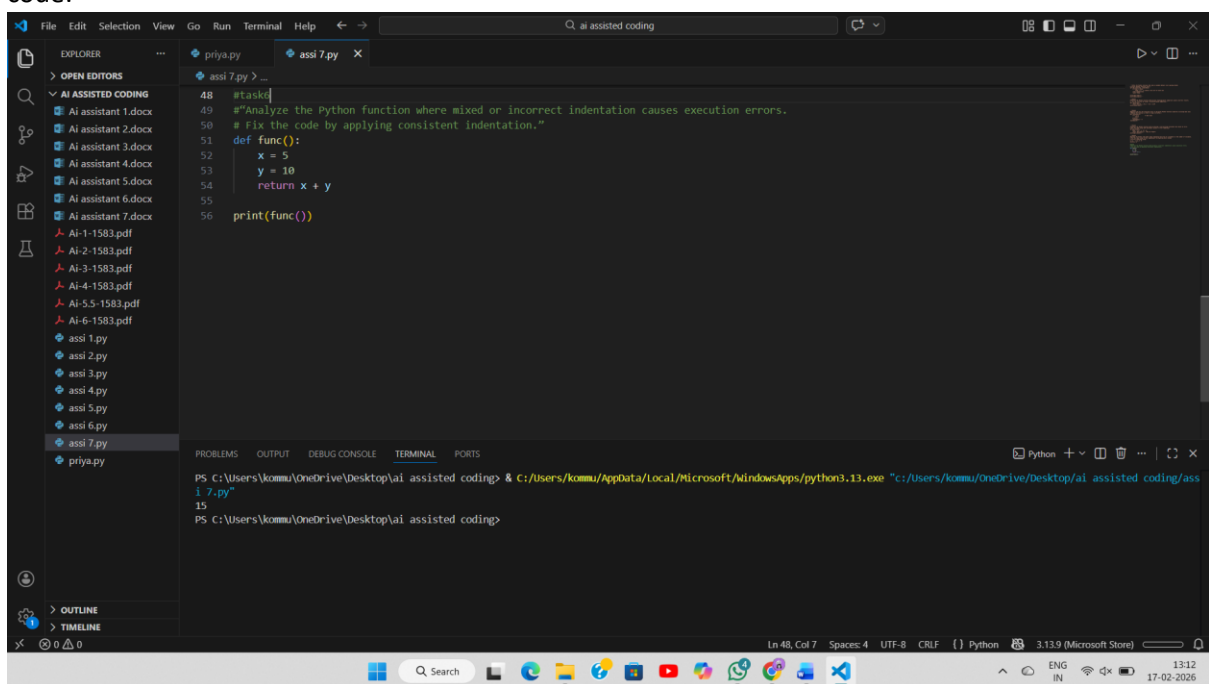
```
def func():
```

```
    x = 5
```

```
    y = 10
```

```
    return x+y
```

Expected Output : Consistent indentation applied.
code:



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

Code:

