Assignment 7.5

Name: **A.JEEVAN SAI**

Ht.no: **2303A51420**

Batch: **21**

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.

Task 1 (Mutable Default Argument – Function Bug)

Task: Analyze given code where a mutable default argument cause unexpected behavior. Use AI to fix it. # Bug: Mutable default argument def add_item(item, items=[]):

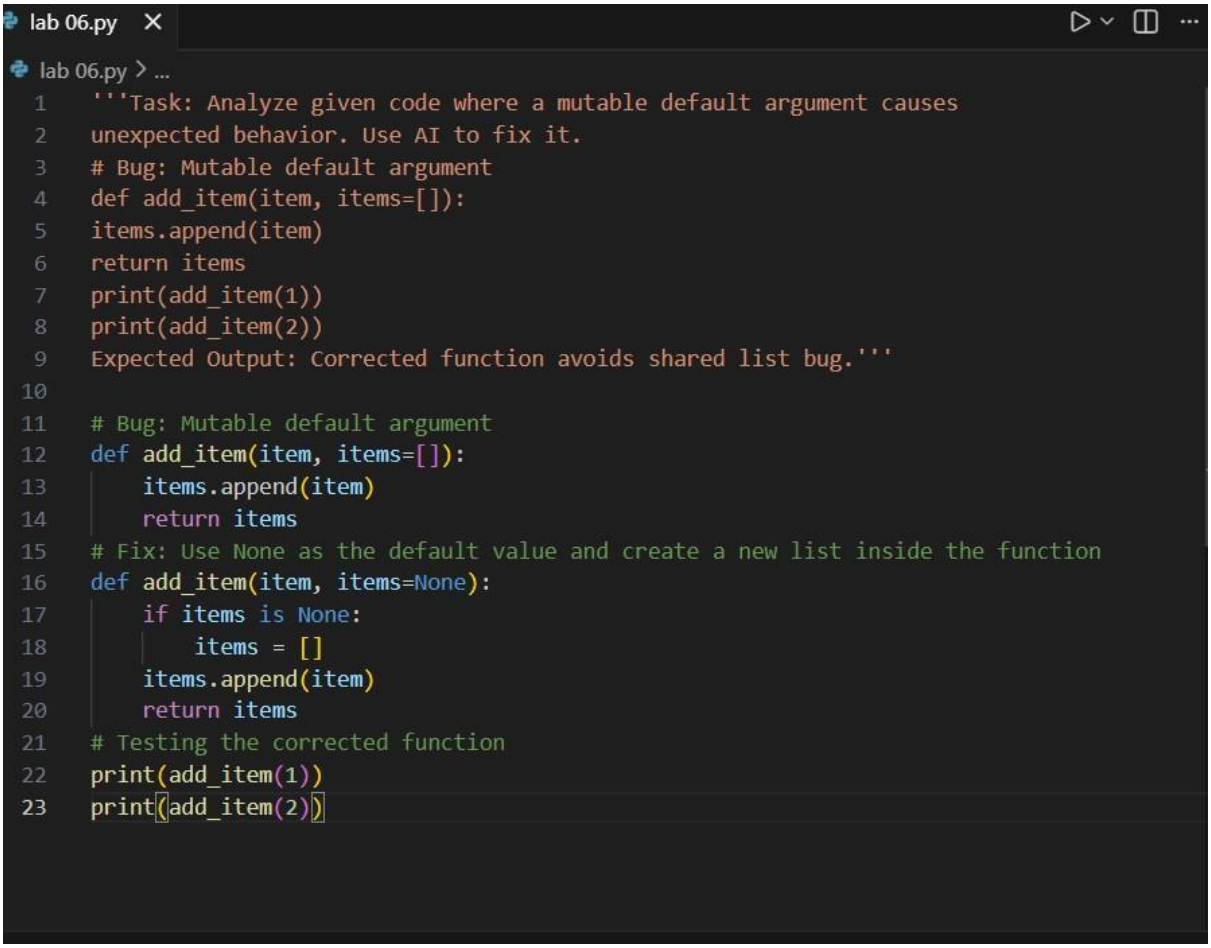
items.append(item) return

items print(add_item(1))

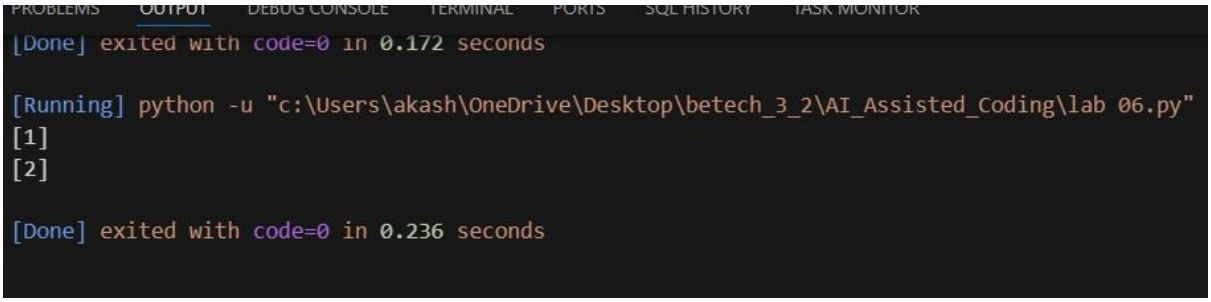
print(add_item(2))

Expected Output: Corrected function avoids shared list bug.

Code:



Output:



Task 2 (Floating-Point Precision Error)

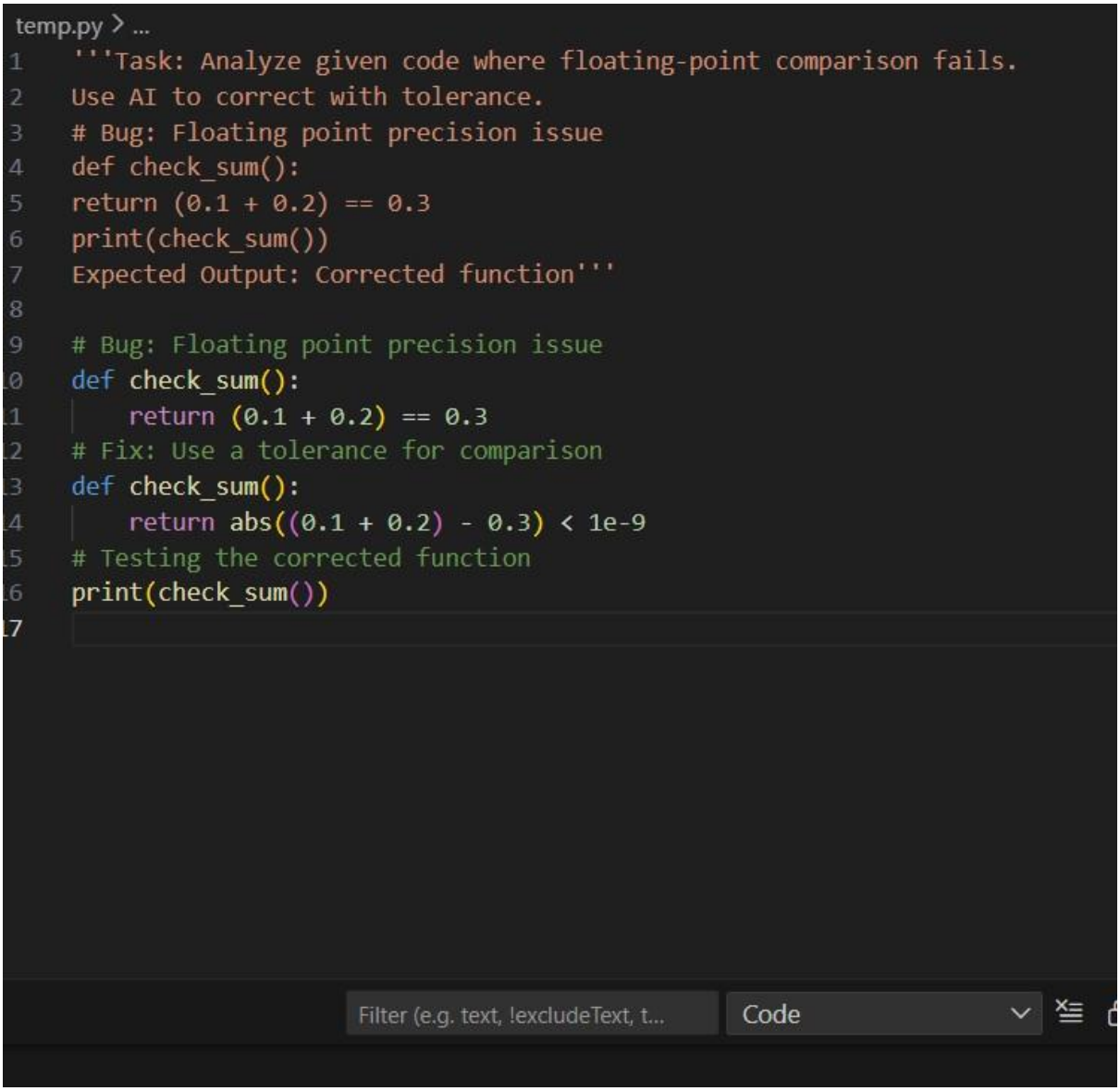
Task: Analyze given code where floating-point comparison fails. Use AI to correct with tolerance. # Bug: Floating point precision issue

def check_sum():

return (0.1 + 0.2) == 0.3
print(check_sum())

Expected Output: Corrected function

Code:



Output:

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)

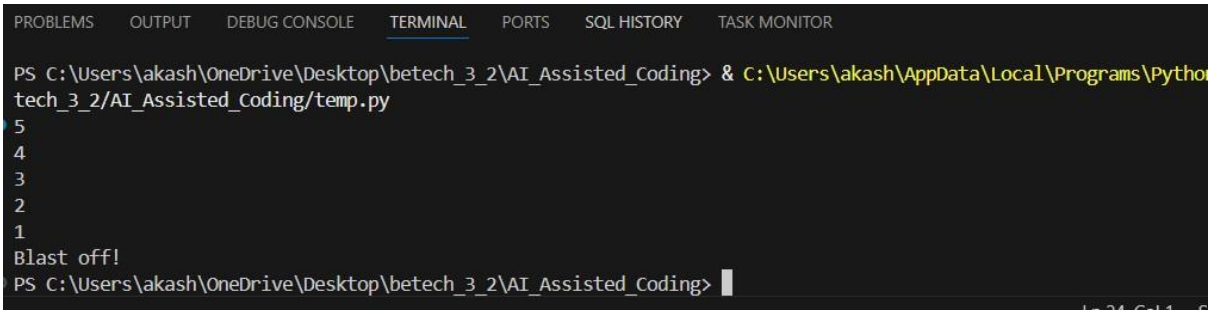
Expected Output : Correct recursion with stopping condition.

Code:

```python
'''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.'''

# Bug: No base case
def countdown(n):
    print(n)
    return countdown(n-1)
# Fix: Add a base case to stop recursion
def countdown(n):
    if n <= 0:
        print("Blast off!")
        return
    print(n)
    return countdown(n-1)
# Testing the corrected function
countdown(5)
```

Output:



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

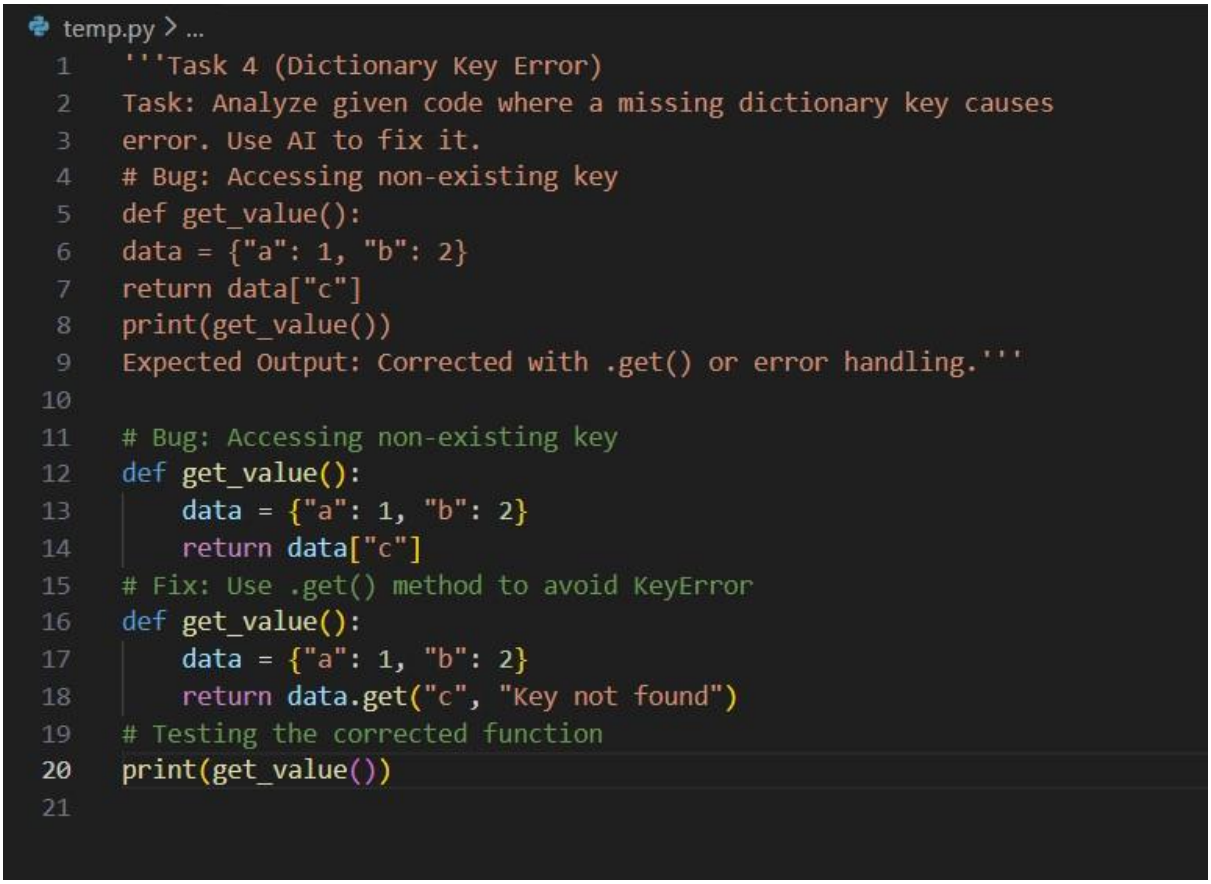
def get_value(): data = {"a": 1, "b":

2} return data["c"]

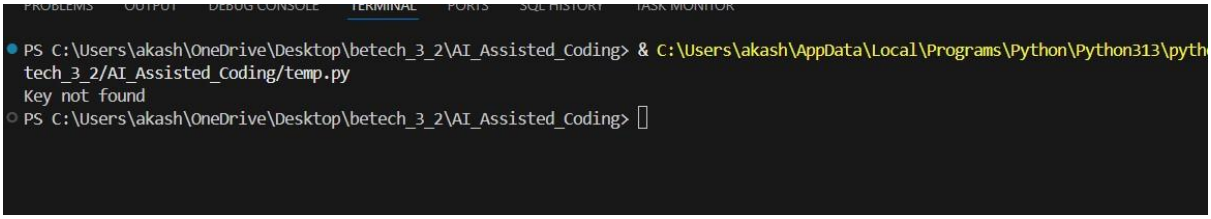
print(get_value())

Expected Output: Corrected with .get() or error handling.

Code:



Output:



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)

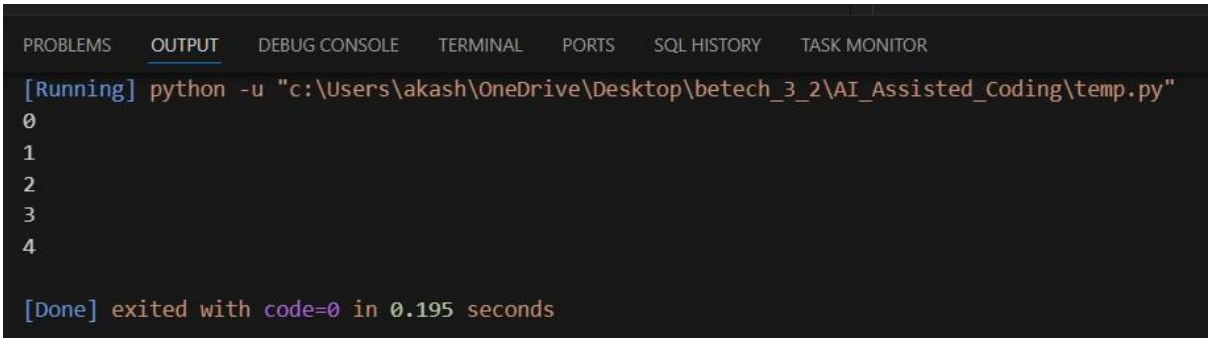
Expected Output: Corrected loop increments i.

Code:

temp.py > loop_example

```
1   '''Task 5 (Infinite Loop – Wrong Condition)
2   Task: Analyze given code where loop never ends. Use AI to detect
3   and fix it.
4   # Bug: Infinite loop
5   def loop_example():
6   i = 0
7   while i < 5:
8   print(i)
9   Expected Output: Corrected loop increments i.'''
10
11  # Bug: Infinite loop
12  def loop_example():
13      i = 0
14      while i < 5:
15          print(i)
16  # Fix: Increment i inside the loop
17  def loop_example():
18      i = 0
19      while i < 5:
20          print(i)
21          i += 1
22  # Testing the corrected function
23  loop_example()
24
25
```

Output:



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:

```
temp.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
8
9   # fix error by correcting the number of variables
10  a, b, c = (1, 2, 3)  # Correcting the number of variables
11  print(a, b, c)
12
13
14
15
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL HISTORY  TASK MONITOR

PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> & C:\Users\akash\AppData\Local\Programs\Pytho
2/AI_Assisted_Coding/temp.py
1 2 3
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\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():

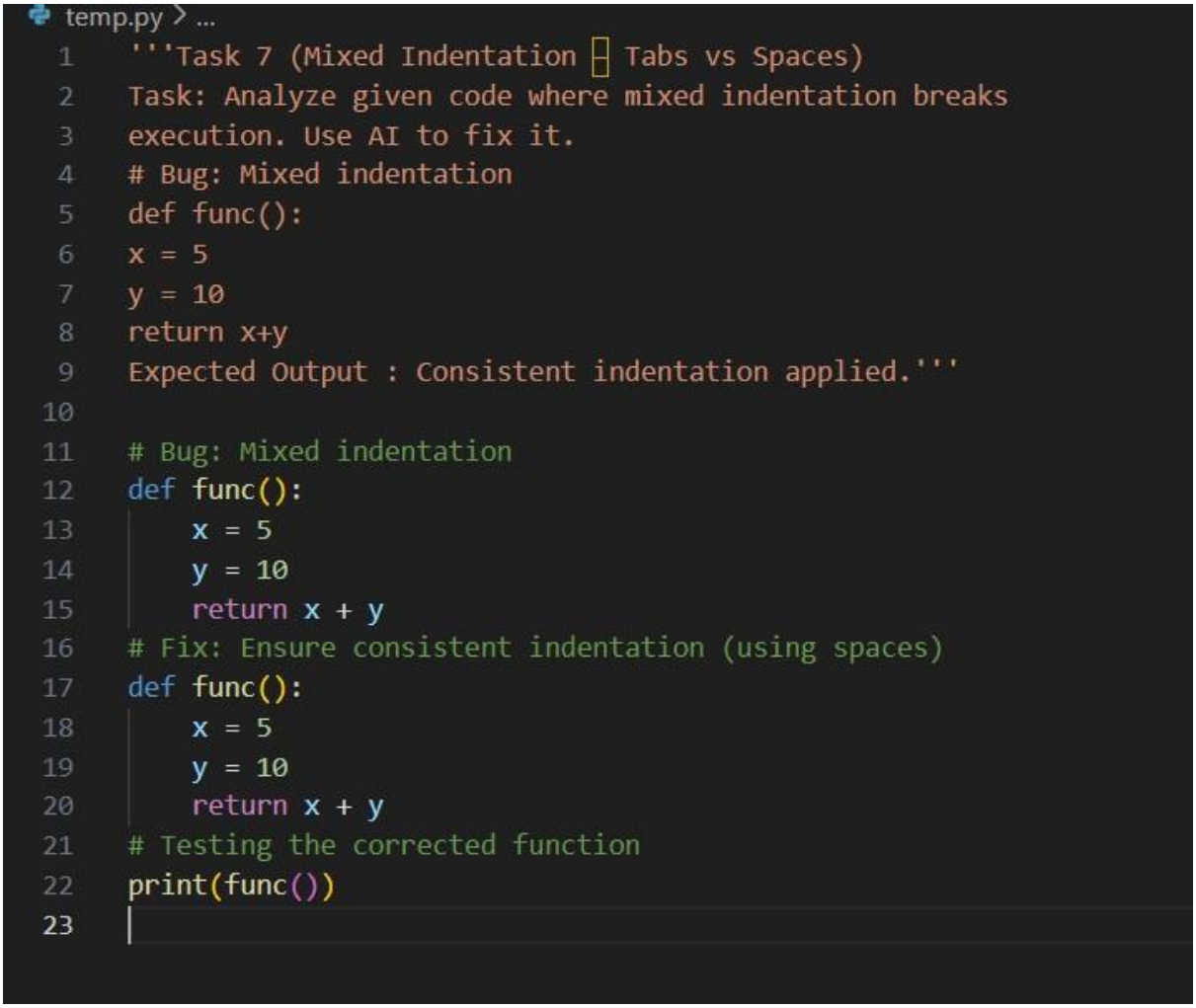
x = 5 y =

10 return

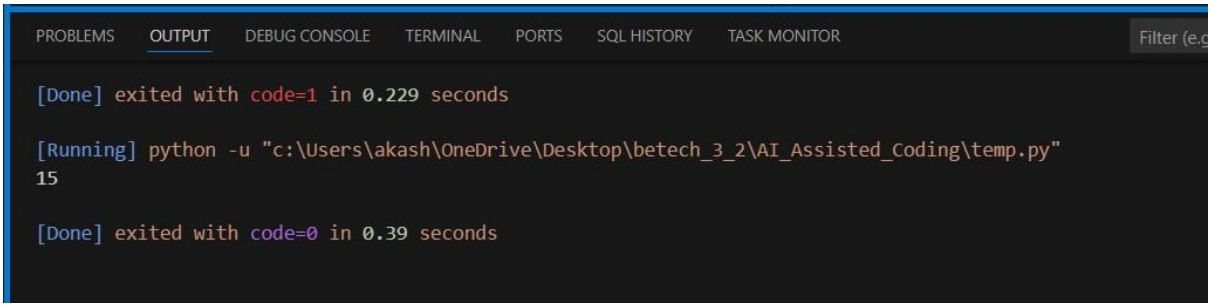
x+y

Expected Output : Consistent indentation applied.

Code:



Output:



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**:

temp.py

```
'''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'''

# Bug: Wrong import
import math
print(math.sqrt(16))
# Fix: Correct the import statement to import math
import math
print(math.sqrt(16))
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL HISTORY  TASK MONITOR
[Done] exited with code=0 in 0.39 seconds

[Running] python -u "c:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding\temp.py"
4.0
4.0

[Done] exited with code=0 in 0.326 seconds
```