AI Assisted Coding

Assignment 7.5

Name: L. Sharan Sai Varshith

Ht.no: 2303A51450

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)

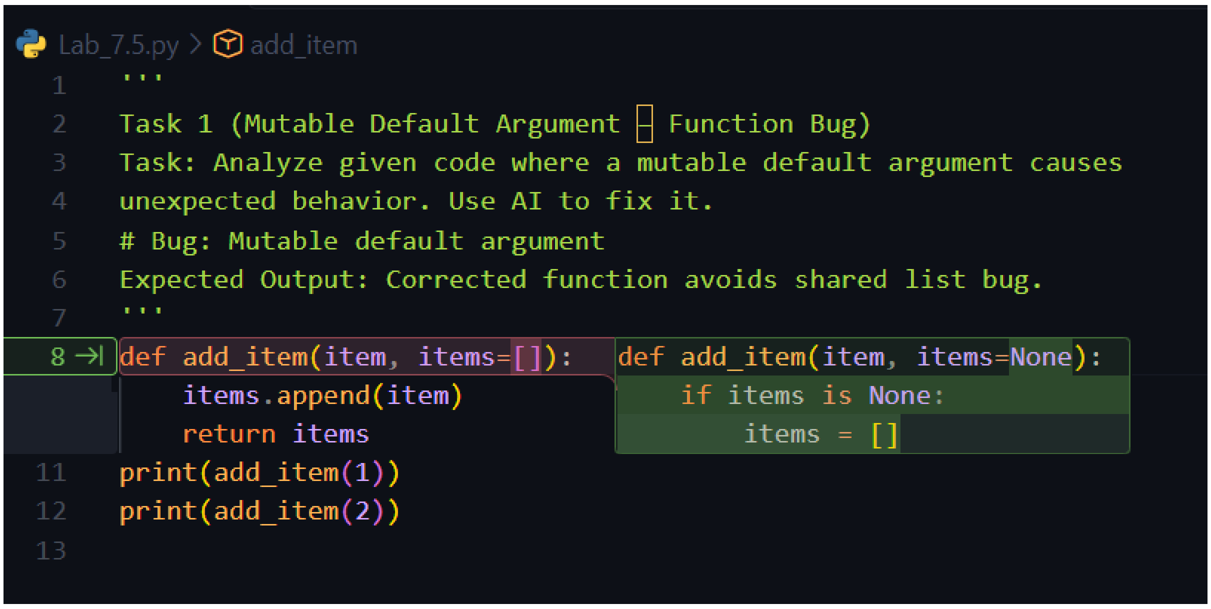
return items

print(add_item(1))

print(add_item(2))

Expected Output: Corrected function avoids shared list bug.

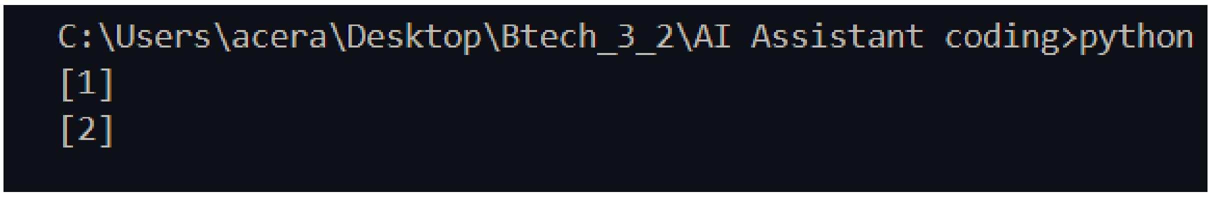
Code:



```
'''
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
Expected Output: Corrected function avoids shared list bug.
'''
def add_item(item, items=[]):
    items.append(item)
    return items
print(add_item(1))
print(add_item(2))
```

```
def add_item(item, items=None):
    if items is None:
        items = []
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python
[1]
[2]
```

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

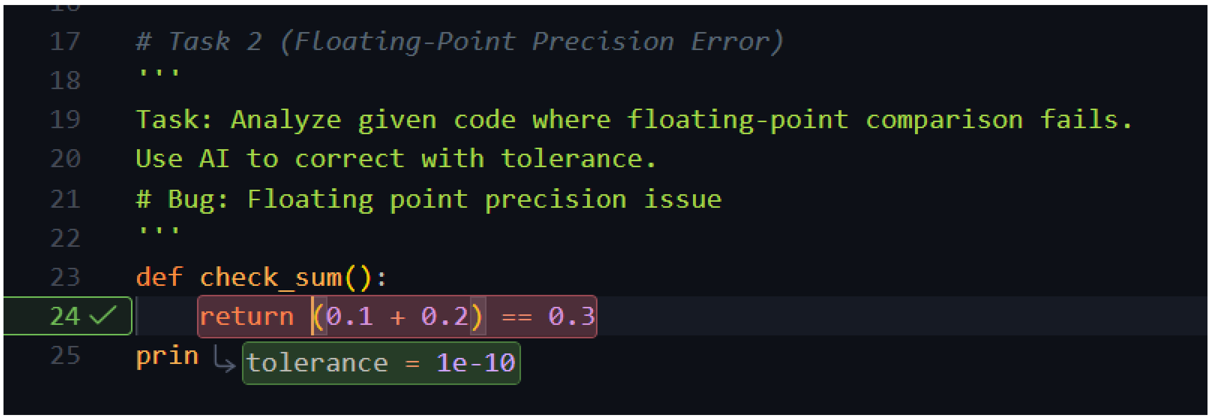
def check_sum():

return (0.1 + 0.2) == 0.3

print(check_sum())

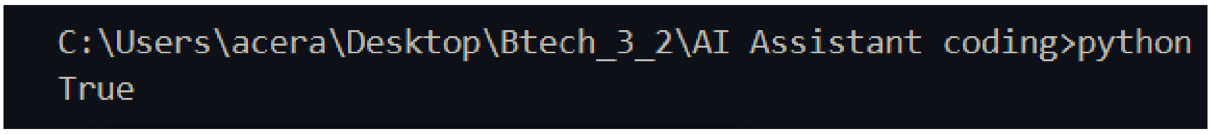
Expected Output: Corrected function

Code:



```
# Task 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
prin ↳ tolerance = 1e-10
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python
True
```

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):

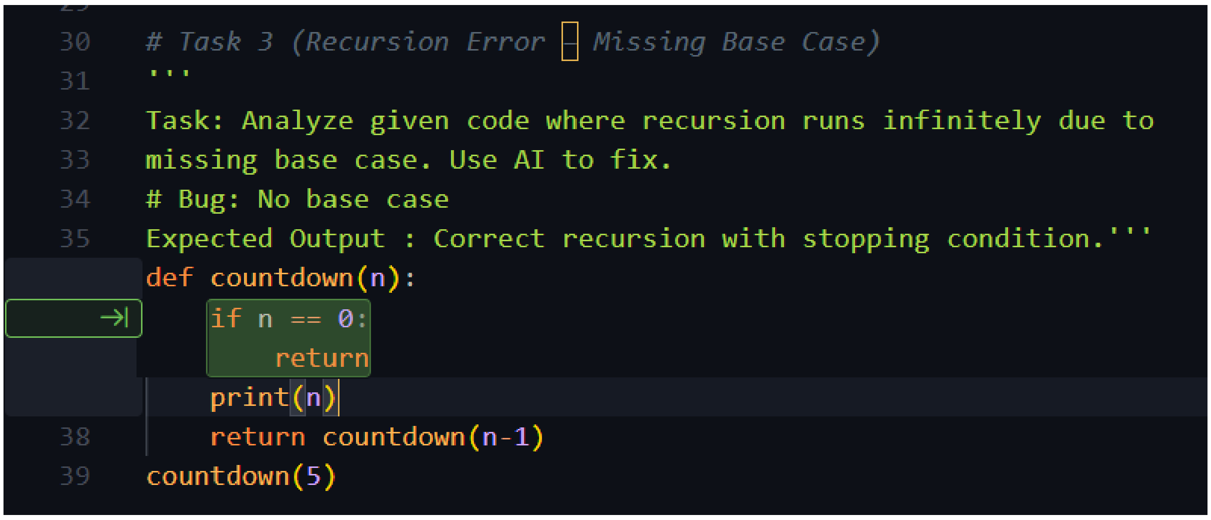
print(n)

return countdown(n-1)

countdown(5)

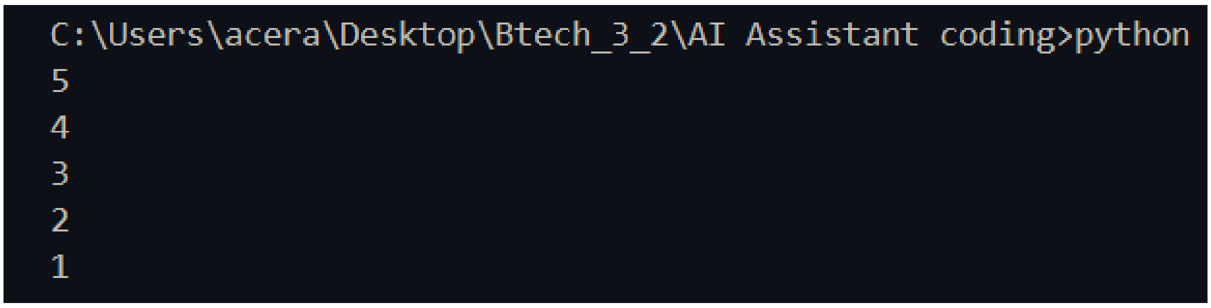
Expected Output : Correct recursion with stopping condition.

Code:



```
# 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
Expected Output : Correct recursion with stopping condition.'''
def countdown(n):
    if n == 0:
        return
    print(n)
    return countdown(n-1)
countdown(5)
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python
5
4
3
2
1
```

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():

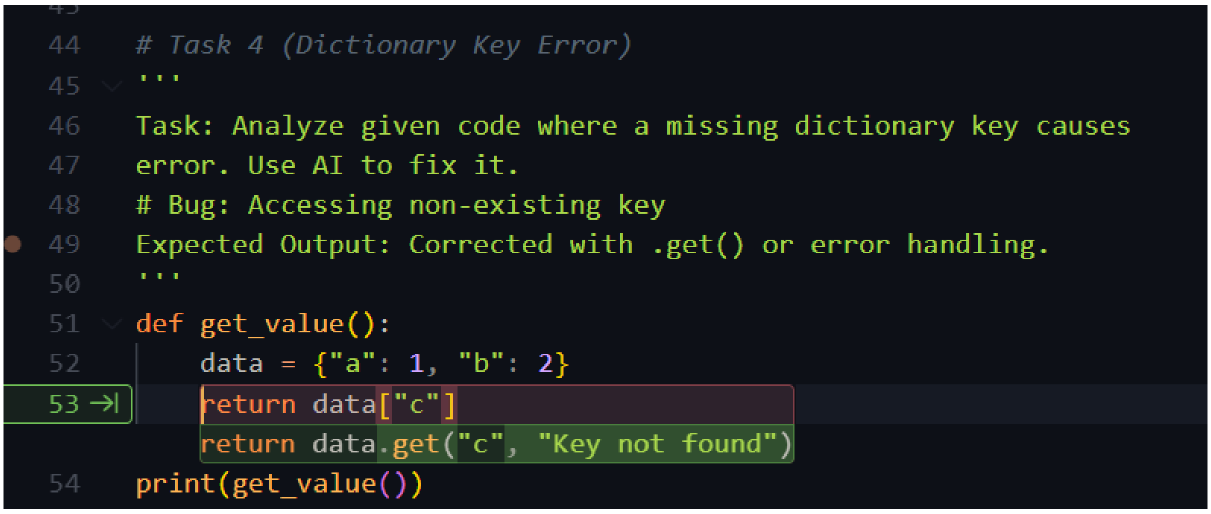
data = {"a": 1, "b": 2}

return data["c"]

print(get_value())

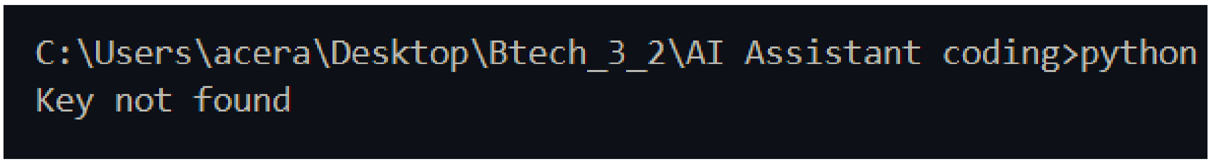
Expected Output: Corrected with .get() or error handling.

Code:



```
# Task 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
Expected Output: Corrected with .get() or error handling.
'''
def get_value():
    data = {"a": 1, "b": 2}
    return data["c"]
    return data.get("c", "Key not found")
print(get_value())
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python
Key not found
```

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():

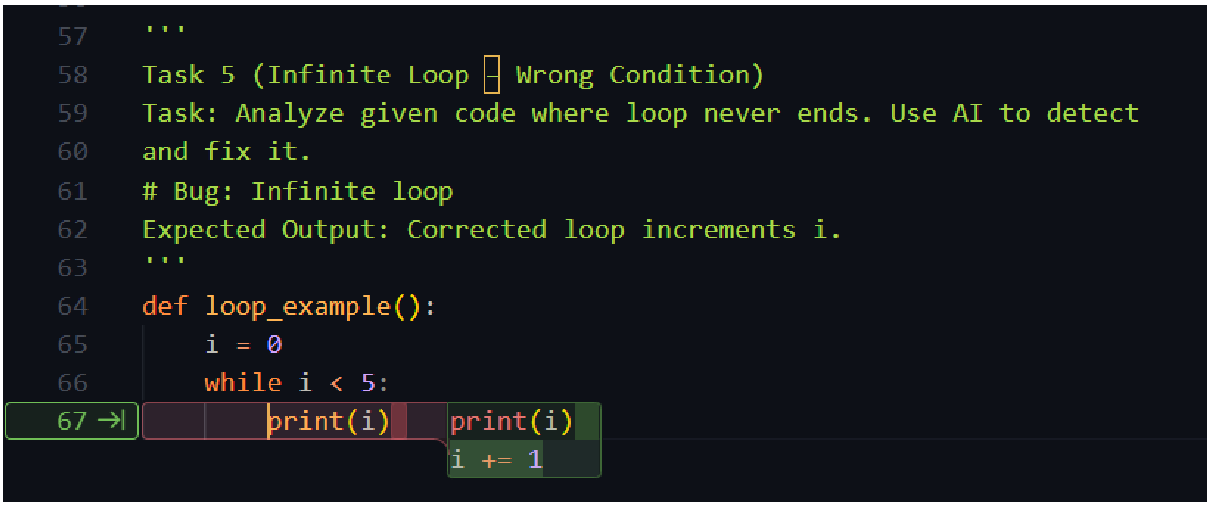
i = 0

while i < 5:

print(i)

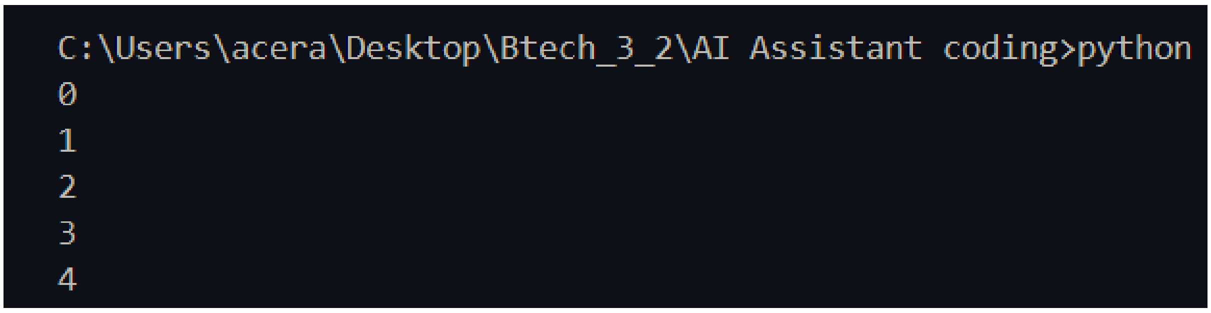
Expected Output: Corrected loop increments i.

Code:



```
'''
Task 5 (Infinite Loop – Wrong Condition)
Task: Analyze given code where loop never ends. Use AI to detect
and fix it.
# Bug: Infinite loop
Expected Output: Corrected loop increments i.
'''
def loop_example():
    i = 0
    while i < 5:
        print(i)    print(i)
                    i += 1
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python
0
1
2
3
4
```

Task 6 (Unpacking Error – Wrong Variables)

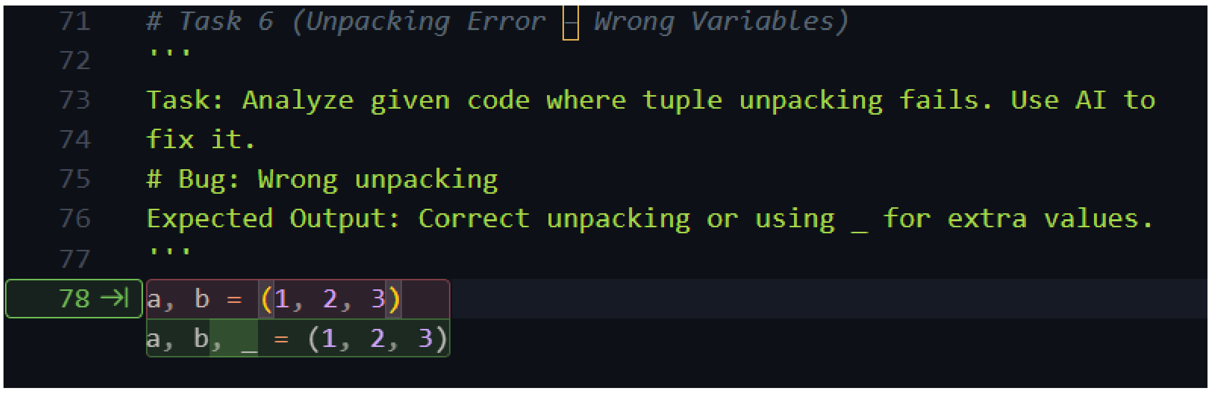
Task: Analyze given code where tuple unpacking fails. Use AI to fix it.

Bug: Wrong unpacking

a, b = (1, 2, 3)

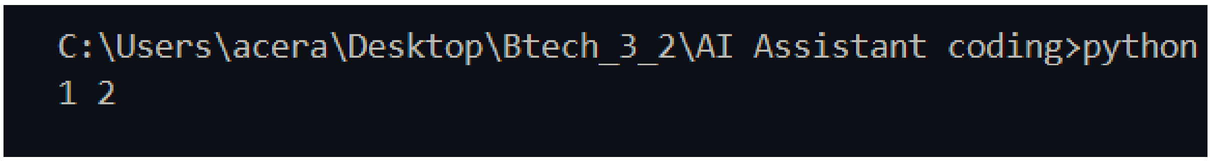
Expected Output: Correct unpacking or using _ for extra values.

Code:



```
# Task 6 (Unpacking Error – Wrong Variables)
'''
Task: Analyze given code where tuple unpacking fails. Use AI to
fix it.
# Bug: Wrong unpacking
Expected Output: Correct unpacking or using _ for extra values.
'''
a, b = (1, 2, 3)
a, b, _ = (1, 2, 3)
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python
1 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():

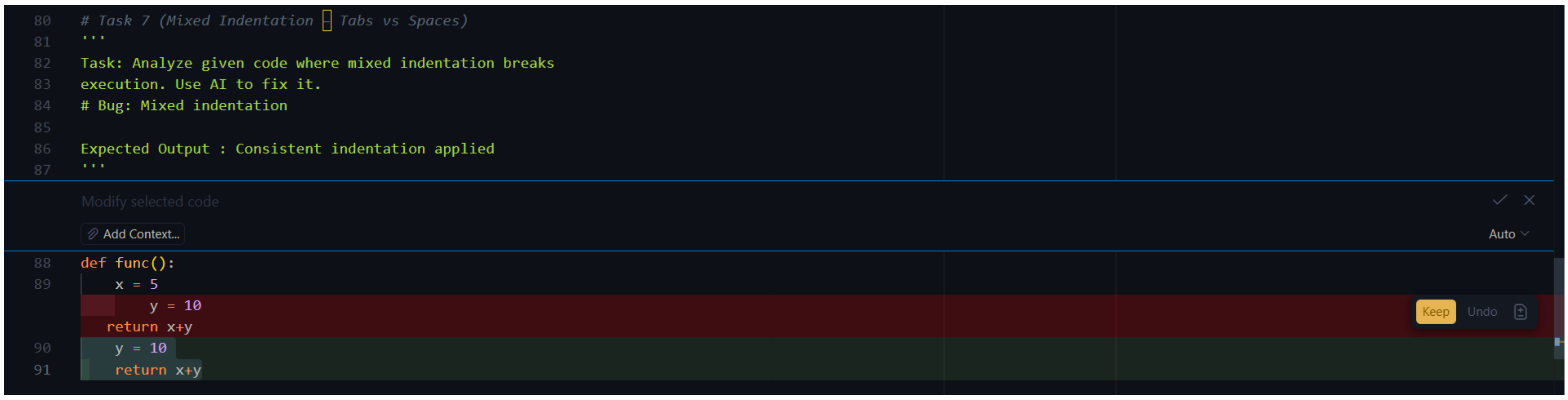
x = 5

y = 10

return x+y

Expected Output : Consistent indentation applied.

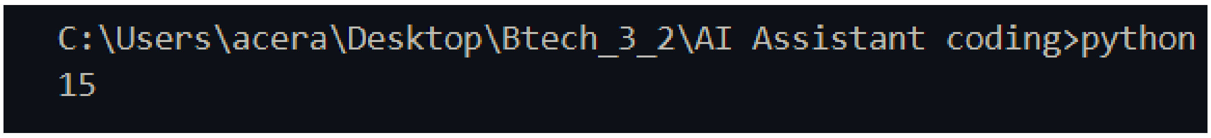
Code:



```
# Task 7 (Mixed Indentation  Tabs vs Spaces)
'''
Task: Analyze given code where mixed indentation breaks
execution. Use AI to fix it.
# Bug: Mixed indentation

Expected Output : Consistent indentation applied
'''
def func():
    x = 5
        y = 10
  return x+y
    y = 10
    return x+y
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python
15
```

Task 8 (Import Error – Wrong Module Usage)

Task: Analyze given code with incorrect import. Use AI to fix.

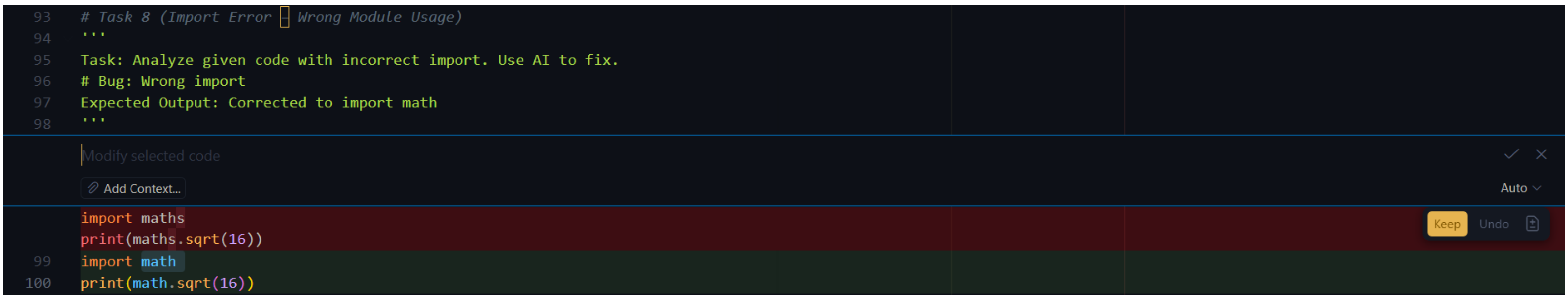
Bug: Wrong import

import maths

print(maths.sqrt(16))

Expected Output: Corrected to import math

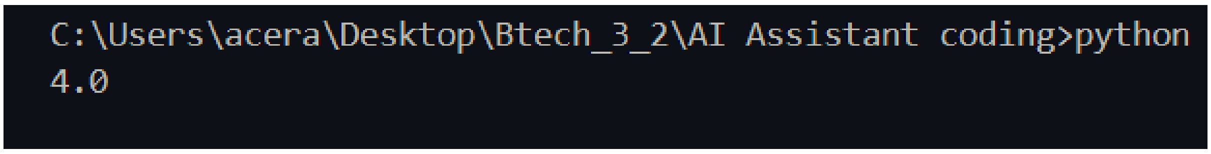
Code:



```
# Task 8 (Import Error – Wrong Module Usage)
'''
Task: Analyze given code with incorrect import. Use AI to fix.
# Bug: Wrong import
Expected Output: Corrected to import math
'''
```

```
import maths
print(maths.sqrt(16))
import math
print(math.sqrt(16))
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python
4.0
```