Lab Assignment-7.5

Name: P.Pooja

Hallticket:2303A510F7

Batch:03

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)

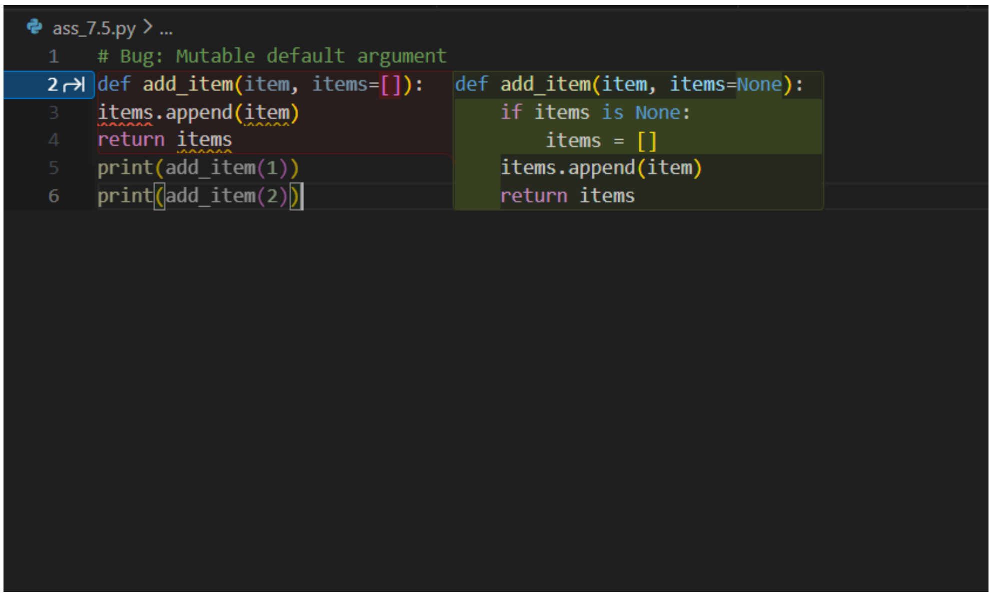
return items

print(add_item(1))

print(add_item(2))

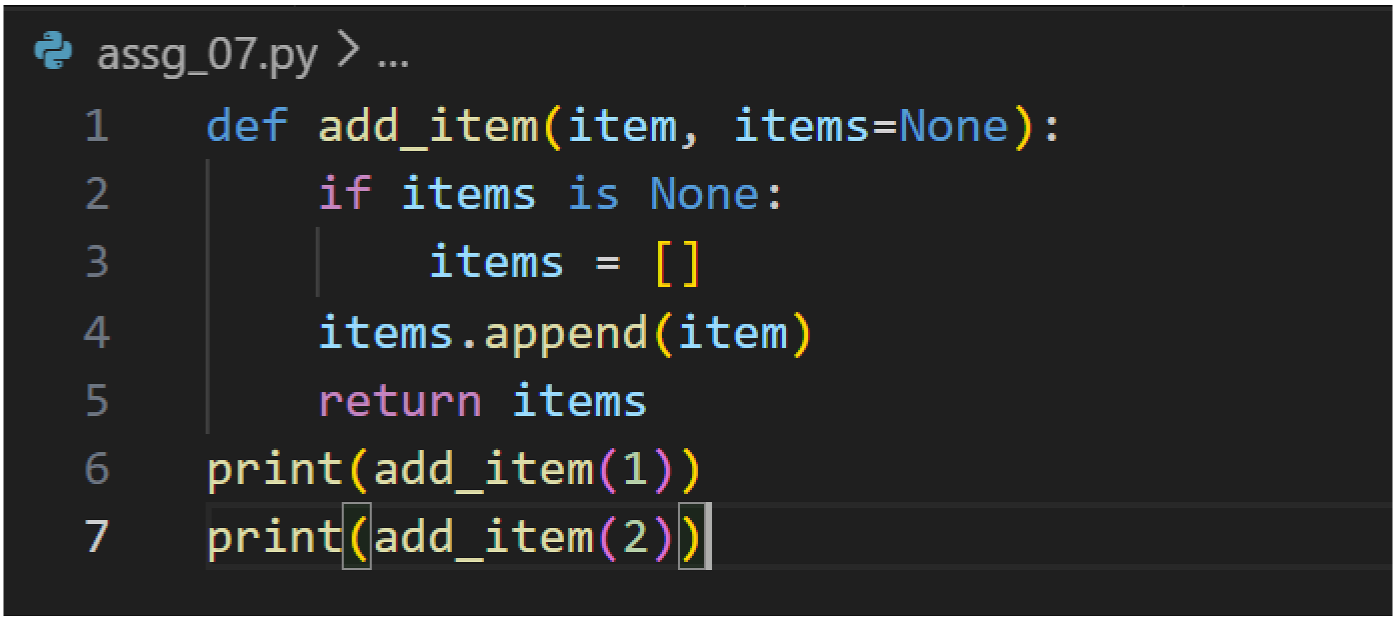
Expected Output: Corrected function avoids shared list bug.

Screenshots:



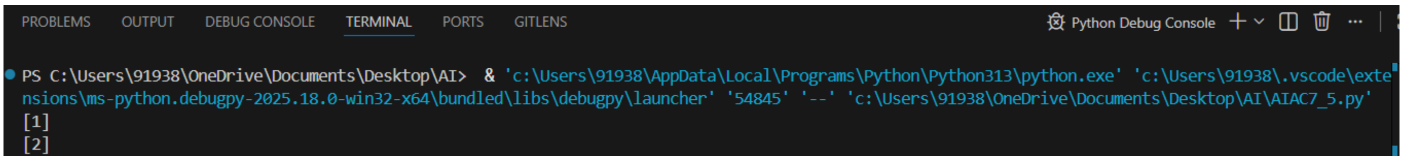
```
# Bug: Mutable default argument
def add_item(item, items=[]):
items.append(item)
return items
print(add_item(1))
print(add_item(2))
```

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



```python
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\91938\OneDrive\Documents\Desktop\AI>  & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\.vscode\exte
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '54845' '--' 'c:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC7_5.py'
[1]
[2]
```

Justification

Each call now gets a **fresh list**, avoiding shared state bugs.

<u>Task 2 (Floating-Point Precision Error)</u>

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

Use AI to correct with tolerance.

# Bug: Floating point precision issue

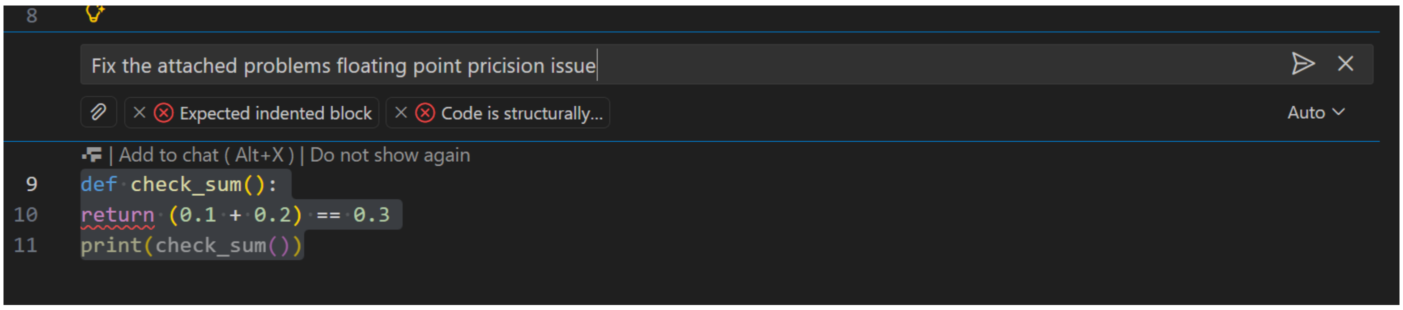
def check_sum():

return (0.1 + 0.2) == 0.3

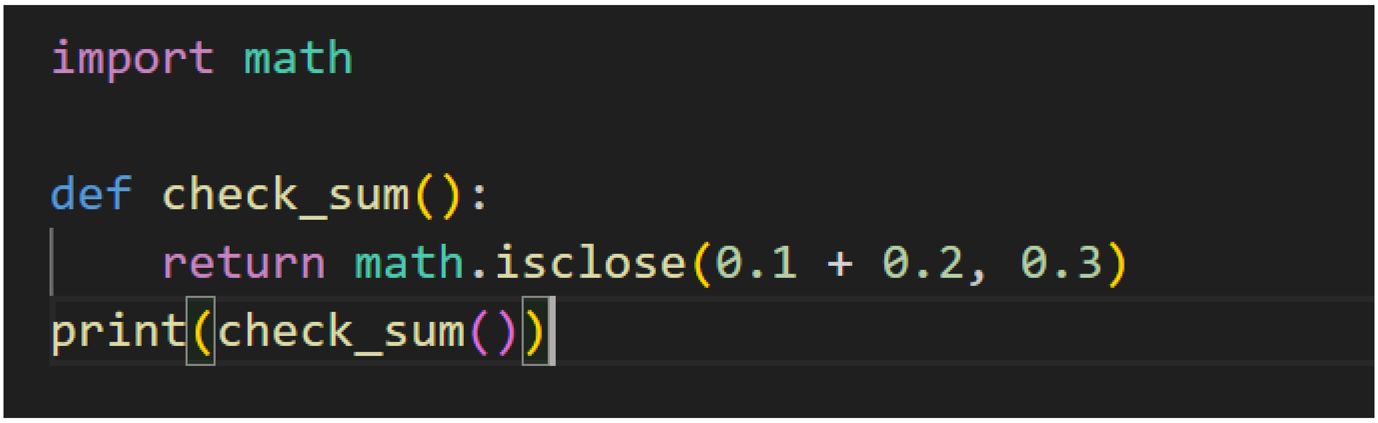
print(check_sum())

Expected Output: Corrected function

Screenshots:



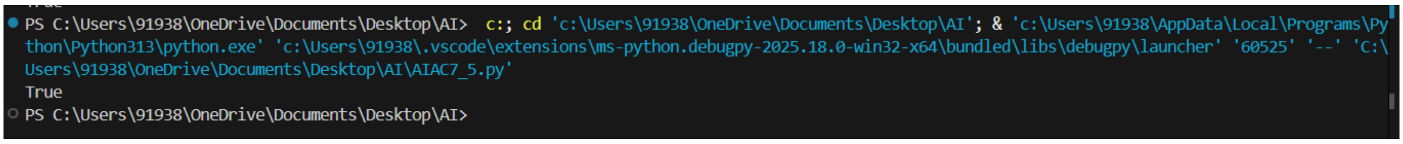
```
Fix the attached problems floating point pricision issue
× Expected indented block   × Code is structurally...   Auto
| Add to chat ( Alt+X ) | Do not show again
def check_sum():
return (0.1 + 0.2) == 0.3
print(check_sum())
```



```python
import math

def check_sum():
    return math.isclose(0.1 + 0.2, 0.3)
print(check_sum())
```

output:



```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>  c:; cd 'c:\Users\91938\OneDrive\Documents\Desktop\AI'; & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60525' '--' 'C:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC7_5.py'
True
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

Justification

Instead of exact equality, we compare within a **small tolerance**, which is the standard way to handle floating-point values.

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):

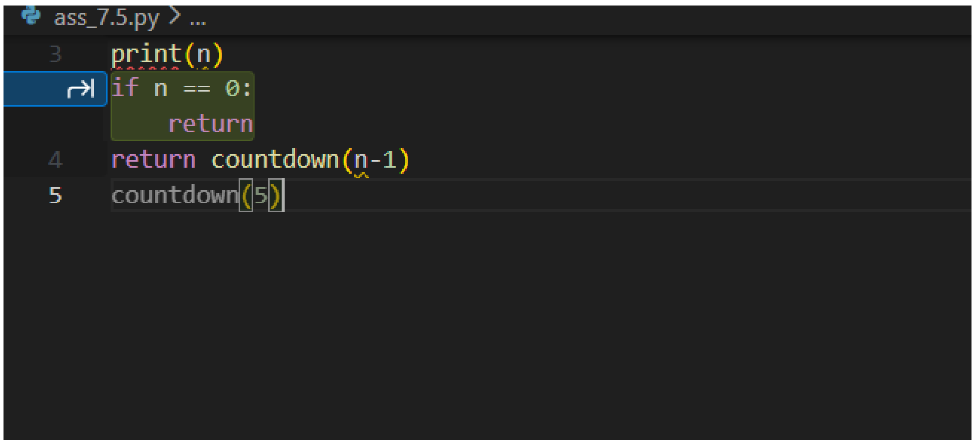
print(n)

return countdown(n-1)

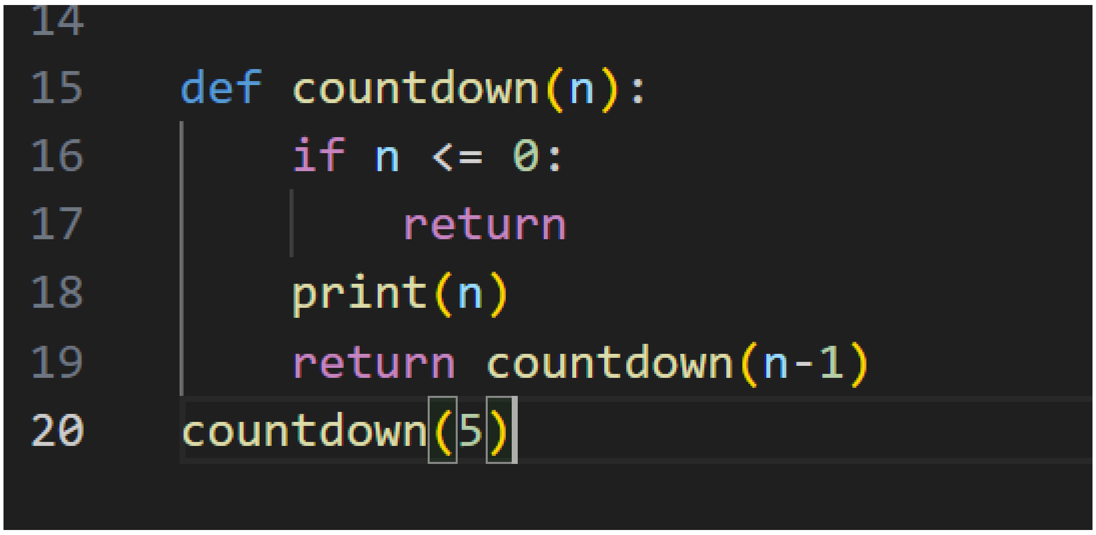
countdown(5)

Expected Output : Correct recursion with stopping condition.

Screenshots:

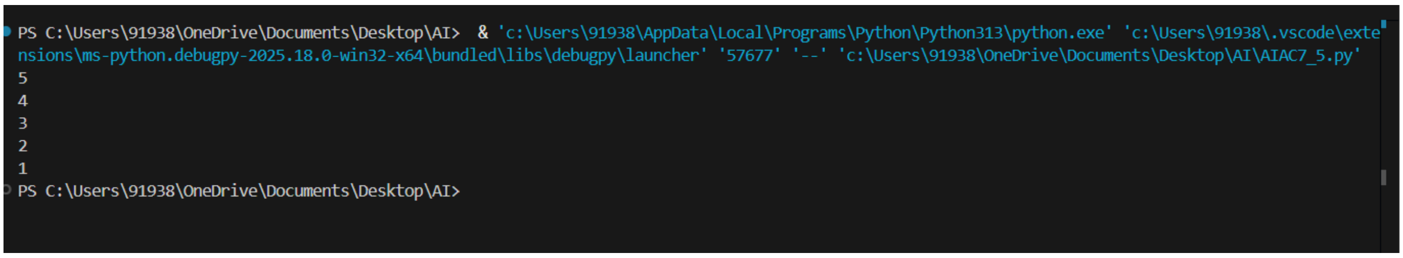


```
ass_7.5.py > ...
print(n)
if n == 0:
    return
return countdown(n-1)
countdown(5)
```



```python
def countdown(n):
    if n <= 0:
        return
    print(n)
    return countdown(n-1)
countdown(5)
```

output:



```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>  & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57677' '--' 'c:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC7_5.py'
5
4
3
2
1
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

Justification

The **base case** (n < 0) stops recursion, preventing infinite calls.

<u>Task 4 (Dictionary Key Error)</u>

Task: Analyze given code where a missing dictionary key causes

error. Use AI to fix it.

# Bug: Accessing non-existing key

def get_value():

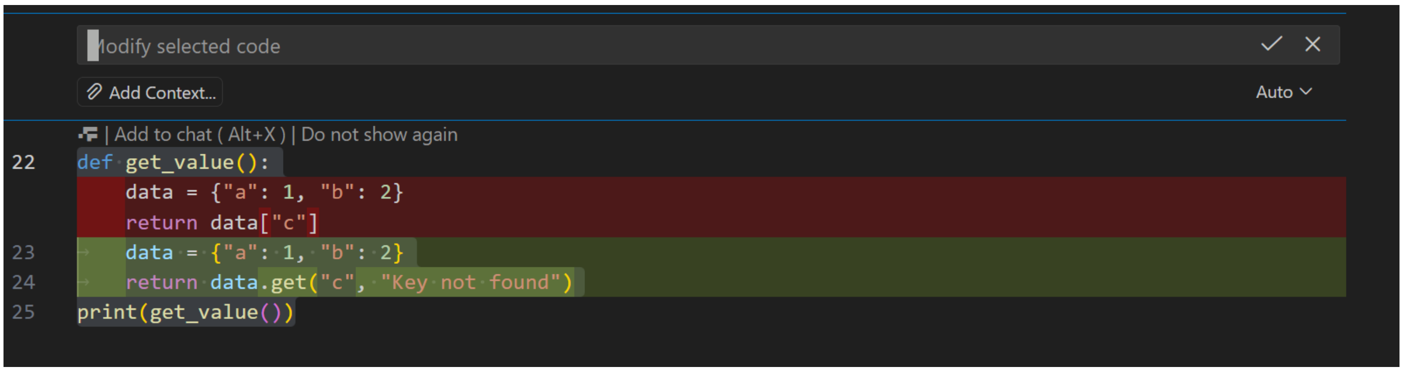
data = {"a": 1, "b": 2}

return data["c"]

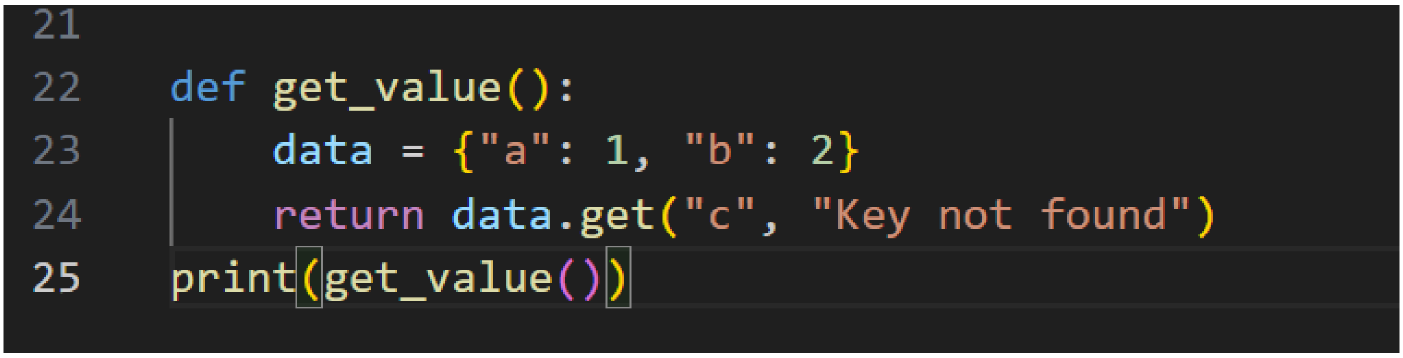
print(get_value())

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

Screenshots:



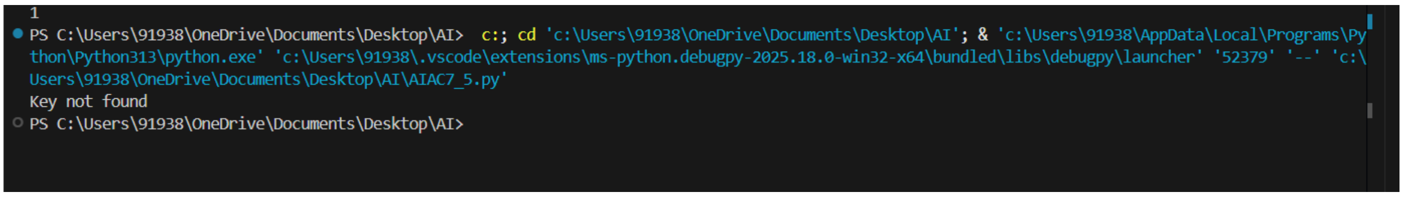
```
Modify selected code
Add Context...                                                    Auto
| Add to chat ( Alt+X ) | Do not show again
def get_value():
    data = {"a": 1, "b": 2}
    return data["c"]
    data = {"a": 1, "b": 2}
    return data.get("c", "Key not found")
print(get_value())
```



```python

def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c", "Key not found")
print(get_value())
```

output:



```
1
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>  c:; cd 'c:\Users\91938\OneDrive\Documents\Desktop\AI'; & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52379' '--' 'c:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC7_5.py'
Key not found
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

Justification

.get() safely handles missing keys and allows a **default value** instead of crashing.

<u>Task 5 (Infinite Loop – Wrong Condition)</u>

Task: Analyze given code where loop never ends. Use AI to detect

and fix it.

# Bug: Infinite loop

def loop_example():

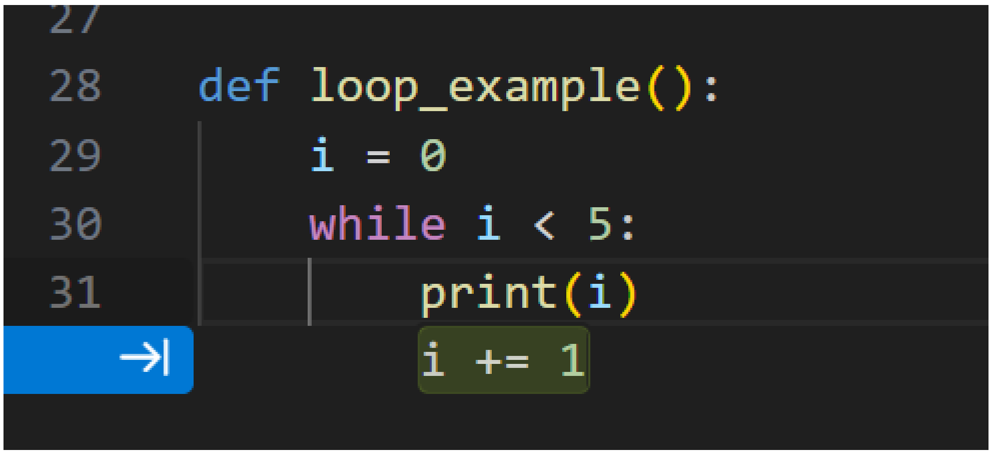
i = 0

while i < 5:

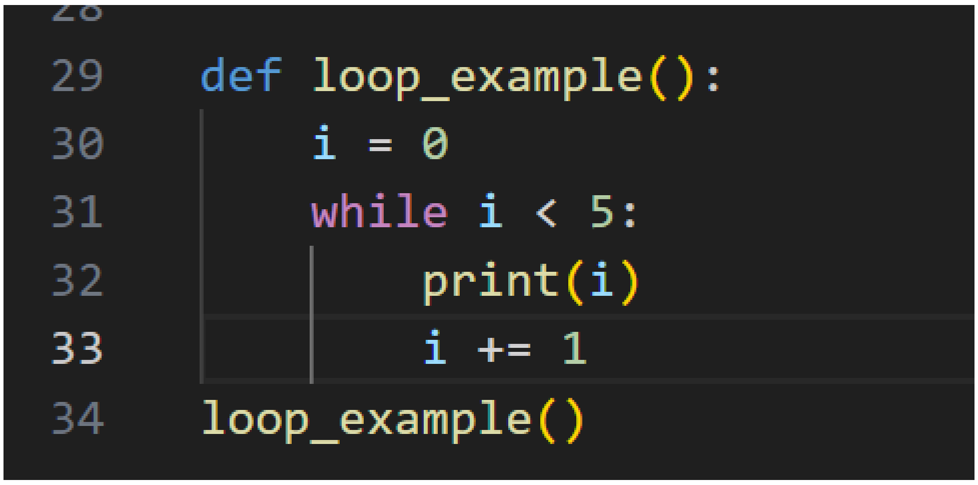
print(i)

Expected Output: Corrected loop increments i.

Screenshots:

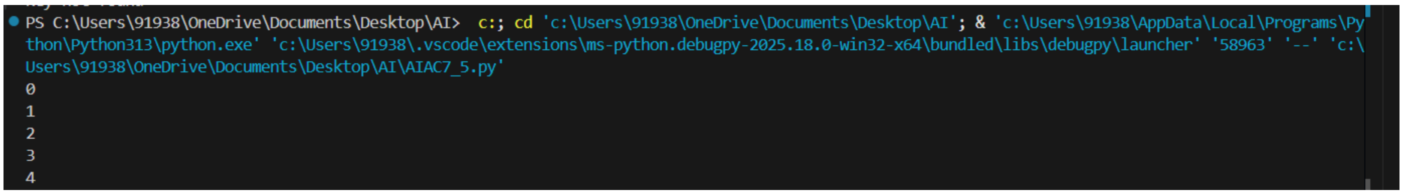


```python
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
```



```python
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
loop_example()
```

output:



```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>  c:; cd 'c:\Users\91938\OneDrive\Documents\Desktop\AI'; & 'c:\Users\91938\AppData\Local\Programs\Py
thon\Python313\python.exe' 'c:\Users\91938\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58963' '--' 'c:\
Users\91938\OneDrive\Documents\Desktop\AI\AIAC7_5.py'
0
1
2
3
4
```

Justification

Updating i ensures the loop condition eventually becomes false.

Task 6 (Unpacking Error – Wrong Variables)

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

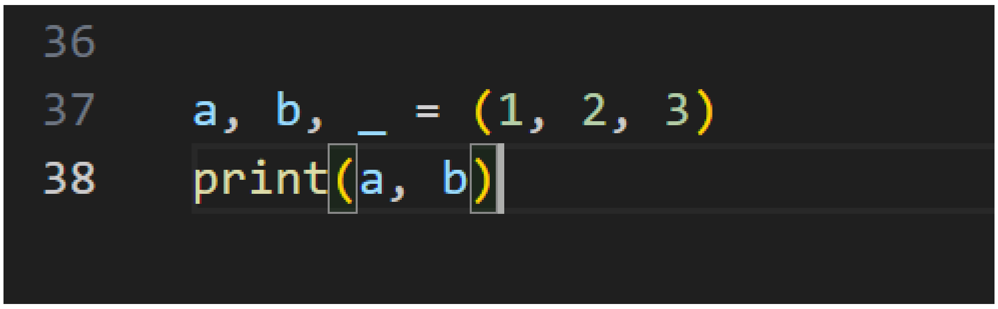
fix it.

# Bug: Wrong unpacking

a, b = (1, 2, 3)

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

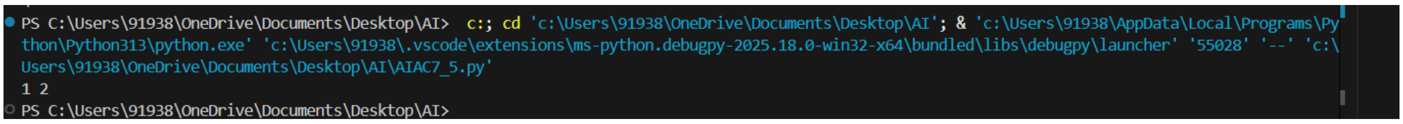
Screenshots:



```python

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

output:



```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>  c:; cd 'c:\Users\91938\OneDrive\Documents\Desktop\AI'; & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55028' '--' 'c:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC7_5.py'
1 2
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

Justification

absorbs extra values, allowing correct unpacking.

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():

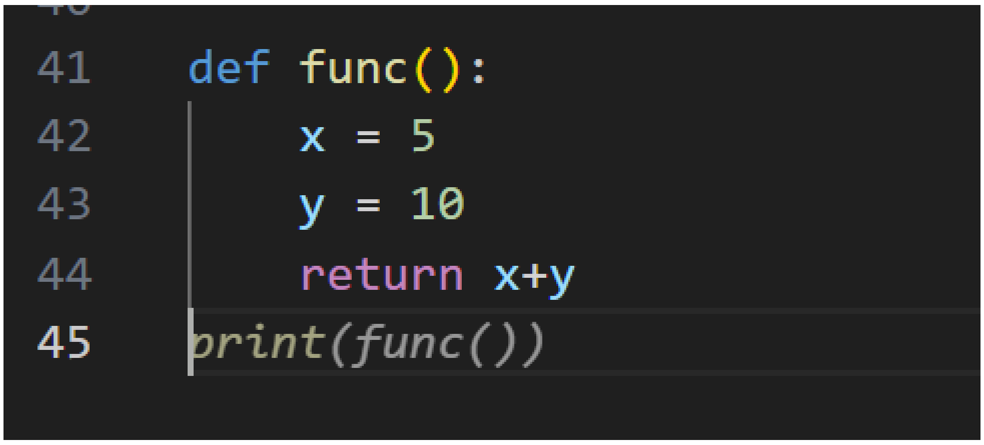
x = 5

y = 10

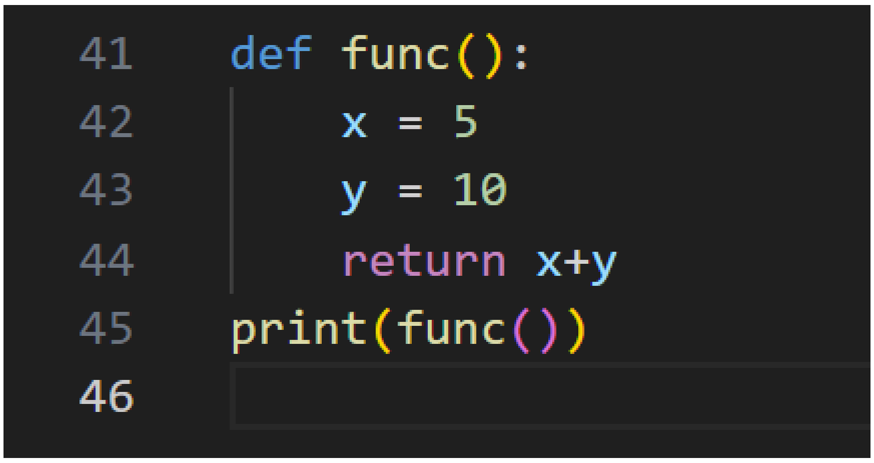
return x+y

Expected Output : Consistent indentation applied.

Screenshots:



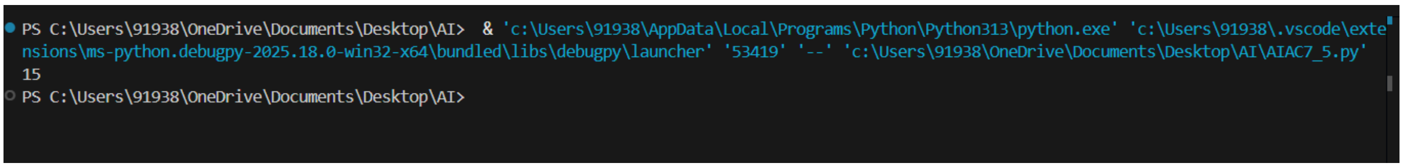
```python
def func():
    x = 5
    y = 10
    return x+y
print(func())
```



```python
def func():
    x = 5
    y = 10
    return x+y
print(func())

```

output:



```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>  & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\.vscode\exte
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53419' '--' 'c:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC7_5.py'
15
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

Justification

Python requires **consistent indentation** (usually 4 spaces).

<u>Task 8 (Import Error – Wrong Module Usage)</u>

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

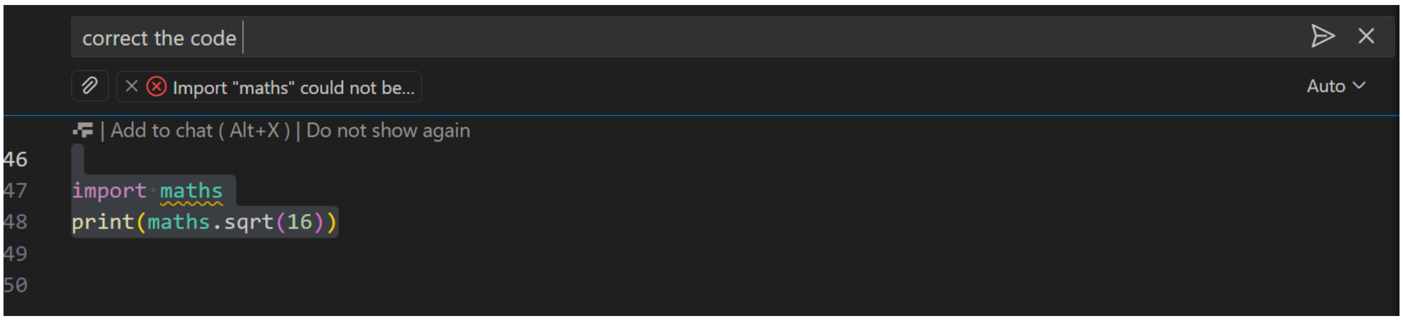
# Bug: Wrong import

import maths

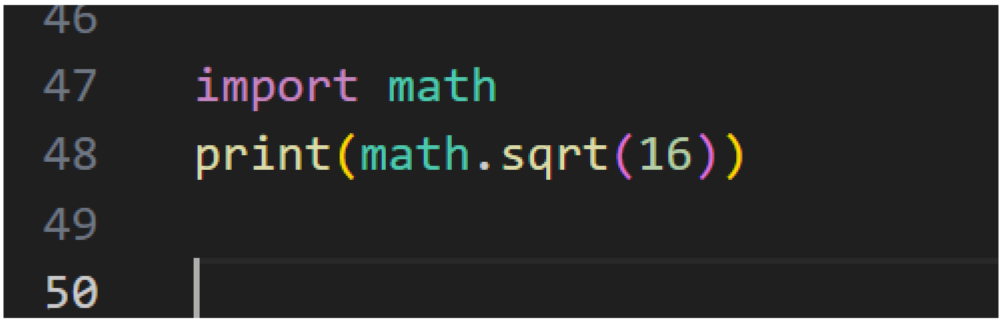
print(maths.sqrt(16))

Expected Output: Corrected to import math

Screenshots:

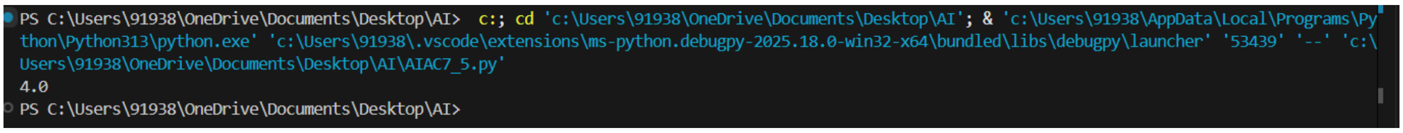

```
correct the code
```
Import "maths" could not be...
Auto
| Add to chat (Alt+X) | Do not show again

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



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

output:



```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>  c:; cd 'c:\Users\91938\OneDrive\Documents\Desktop\AI'; & 'c:\Users\91938\AppData\Local\Programs\Py
thon\Python313\python.exe' 'c:\Users\91938\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53439' '--' 'c:\
Users\91938\OneDrive\Documents\Desktop\AI\AIAC7_5.py'
4.0
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

Justification

The correct standard library module is math, not maths.