AI ASSISTED CODING

LAB-7.5

gayathri

2303A51909

Batch-12

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=[]):

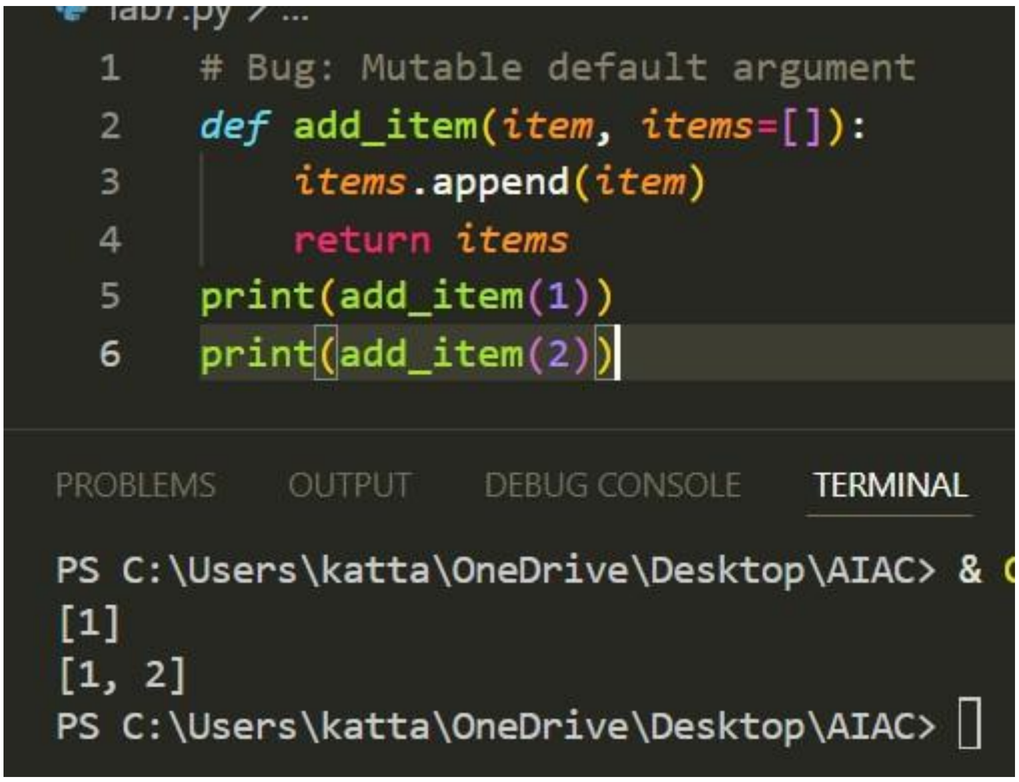
items.append(item) return

items print(add_item(1))

print(add_item(2))

Expected Output: Corrected function avoids shared list bug.

Given Code and Output:



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

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C
[1]
[1, 2]
PS C:\Users\katta\OneDrive\Desktop\AIAC>
```

Task 2 (Floating-Point Precision Error)

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

Use AI to correct with tolerance. #

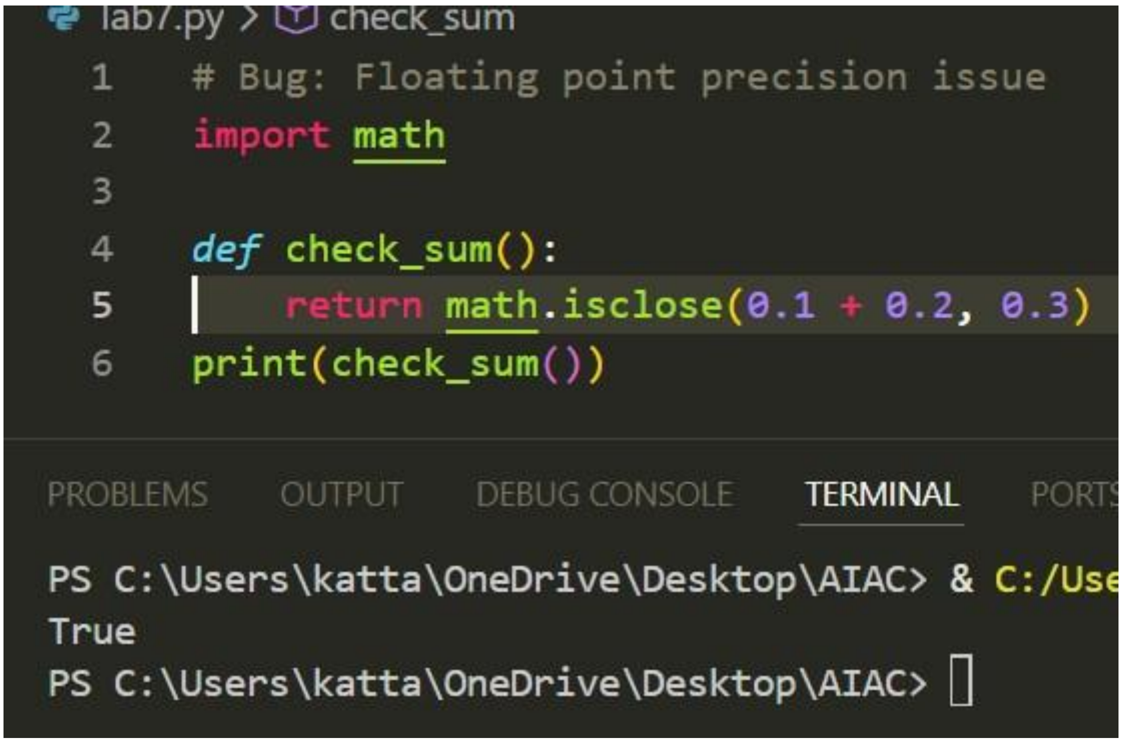
Bug: Floating point precision issue

def check_sum(): return (0.1 + 0.2)

== 0.3 print(check_sum())

Expected Output: Corrected function <u>Given</u>

<u>Code and Output:</u>



```
lab7.py > check_sum
# Bug: Floating point precision issue
import math

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

```
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C:/Use
True
PS C:\Users\katta\OneDrive\Desktop\AIAC>
```

<u>Task 3: (Recursion Error – Missing Base Case)</u>

Task: Analyze given code where recursion runs infinitely due to

missing base case. Use AI to fix.

# Bug: No base case def

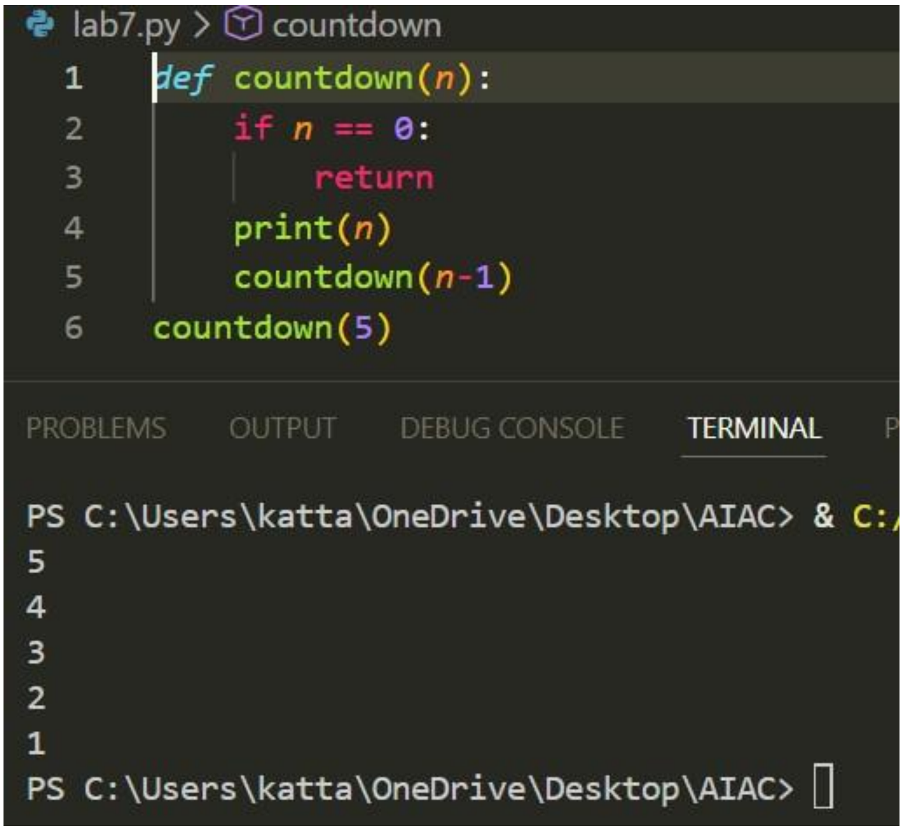
countdown(n):

print(n)

return countdown(n-1) countdown(5)

Expected Output : Correct recursion with stopping condition.

<u>Given Code and Output:</u>


lab7.py > countdown

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C:
5
4
3
2
1
PS C:\Users\katta\OneDrive\Desktop\AIAC>
```

<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

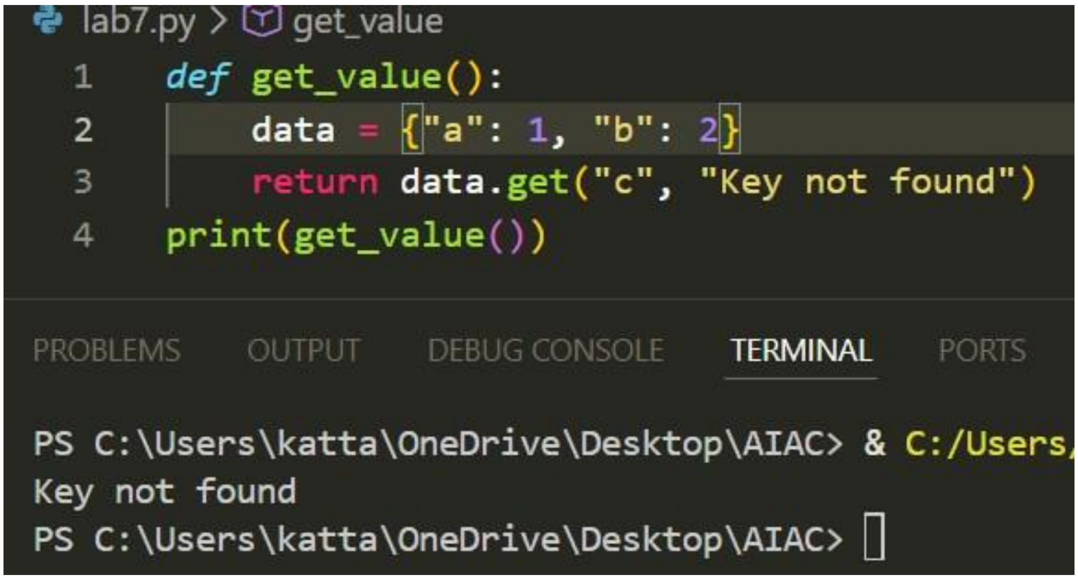
def get_value(): data = {"a": 1,

"b": 2} return data["c"]

print(get_value())

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

<u>Given Code and output:</u>



lab7.py > get_value

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C:/Users
Key not found
PS C:\Users\katta\OneDrive\Desktop\AIAC>
```

<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():

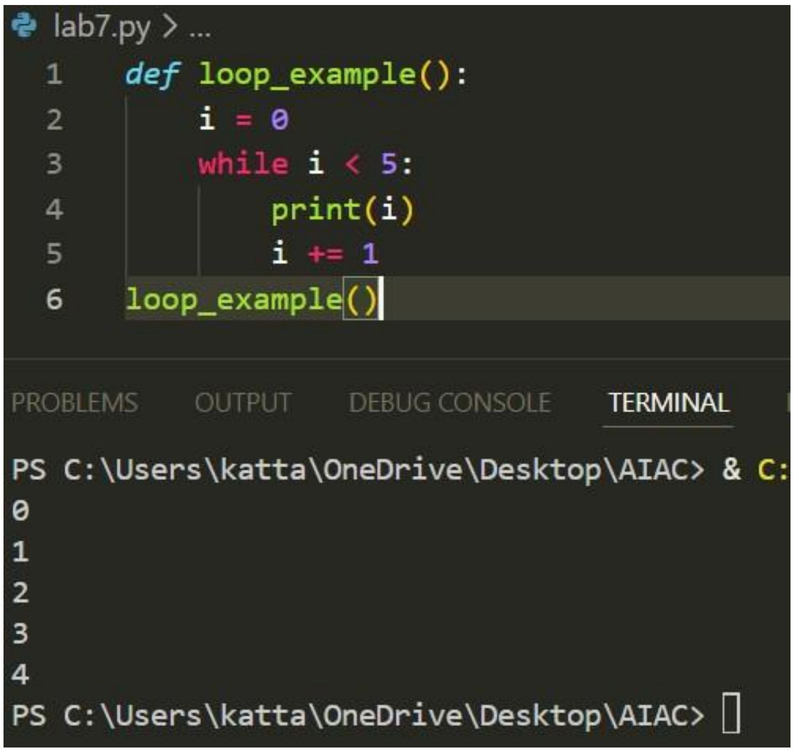
i = 0 while

i < 5:

print(i)

Expected Output: Corrected loop increments i.

<u>Given Code and Output:</u>



```
lab7.py > ...
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
loop_example()
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C:
0
1
2
3
4
PS C:\Users\katta\OneDrive\Desktop\AIAC>
```

<u>Task 6 (Unpacking Error – Wrong Variables)</u>

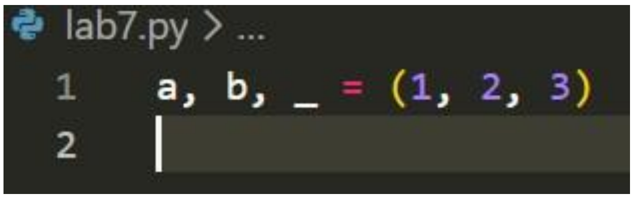
Task: Analyze given code where tuple unpacking fails. Use AI to fix it.

# Bug: Wrong unpacking

a, b = (1, 2, 3)

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

<u>Given Code:</u>



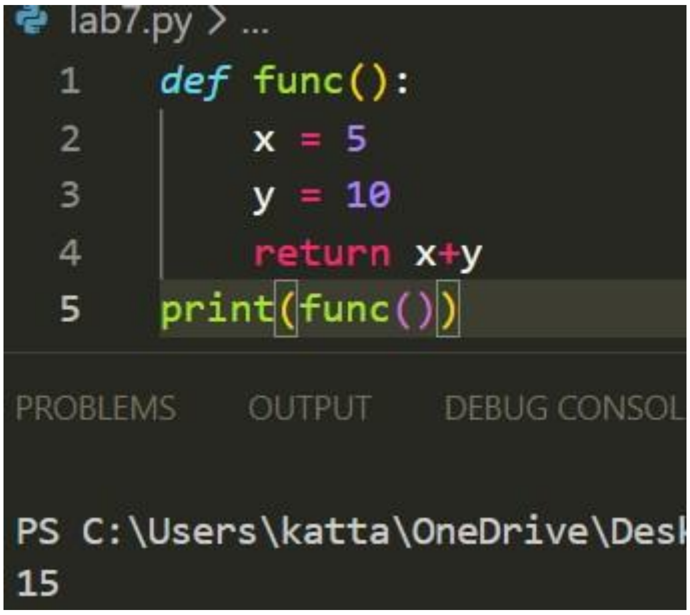
```
lab7.py > ...
a, b, _ = (1, 2, 3)
```

<u>Task 7 (Mixed Indentation – Tabs vs Spaces)</u>

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.

<u>Given Code and Output:</u>



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

```
PS C:\Users\katta\OneDrive\Desk
15
```

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

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

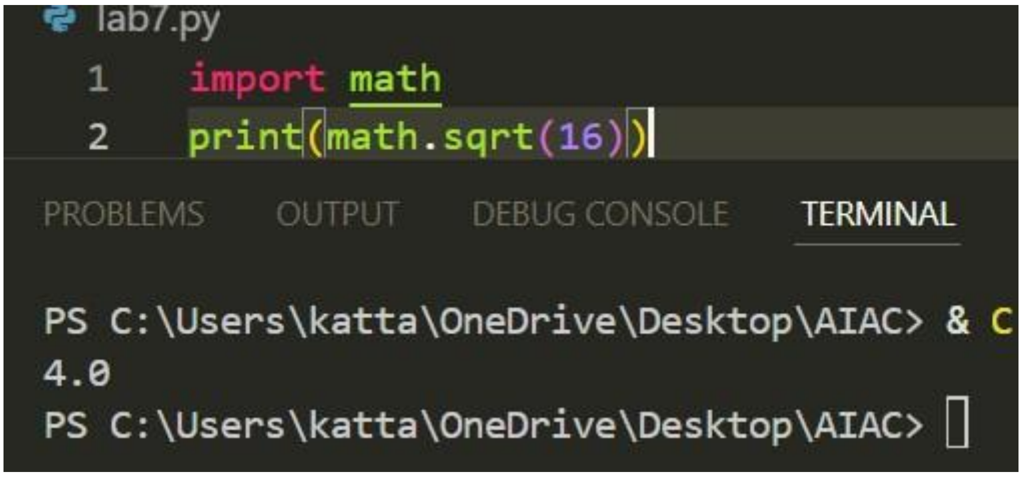
# Bug: Wrong import

import maths

print(maths.sqrt(16))

Expected Output: Corrected to import math <u>Given</u>

<u>**Code and Output:**</u>



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

```
PS C:\Users\katta\OneDrive\Desktop\AIAC> & C
4.0
PS C:\Users\katta\OneDrive\Desktop\AIAC>
```