Assignment Number: 7.5

NAME: N. Nanda Mukesh

H.T.NO: 2303A51410

BATCH: 24

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=[]):

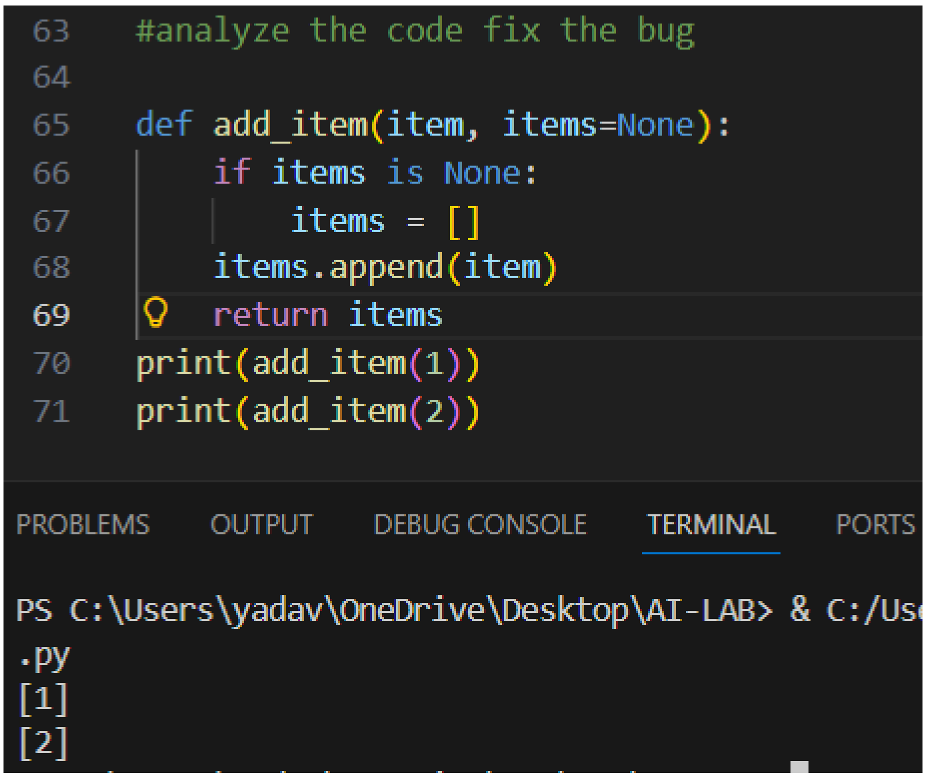
items.append(item)

return items

print(add_item(1))

print(add_item(2))

Expected Output: Corrected function avoids shared list bug.



```python
#analyze the code fix the bug

def add_item(item, items=None):
    if items is None:
        items = []
    items.append(item)
    return items
print(add_item(1))
print(add_item(2))
```

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> & C:/Us
.py
[1]
[2]
```

Task 2 (Floating-Point Precision Error)

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

Use AI to correct with tolerance.

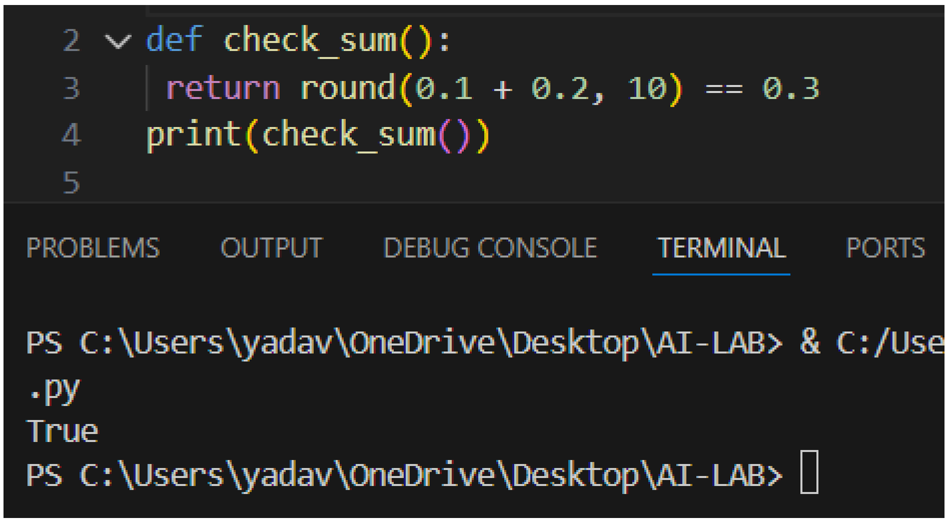
# Bug: Floating point precision issue

def check_sum():

return (0.1 + 0.2) == 0.3

print(check_sum())

Expected Output: Corrected function



```python
def check_sum():
    return round(0.1 + 0.2, 10) == 0.3
print(check_sum())
```

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> & C:/Use
.py
True
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> 
```

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

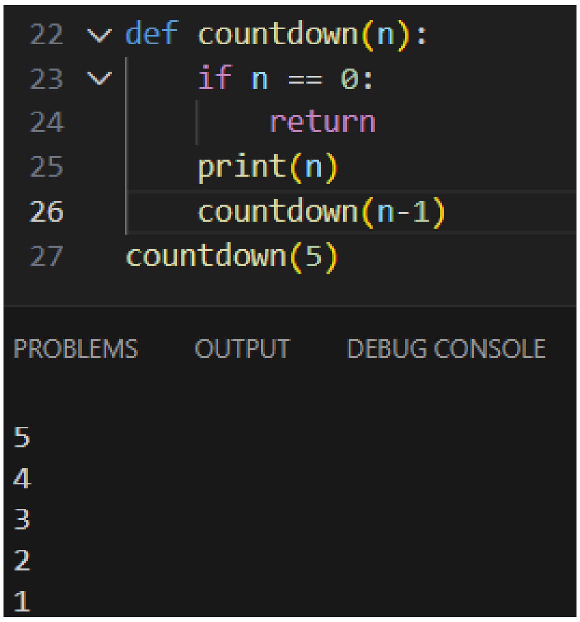
def countdown(n):

print(n)

return countdown(n-1)

countdown(5)

Expected Output : Correct recursion with stopping condition.



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

```
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

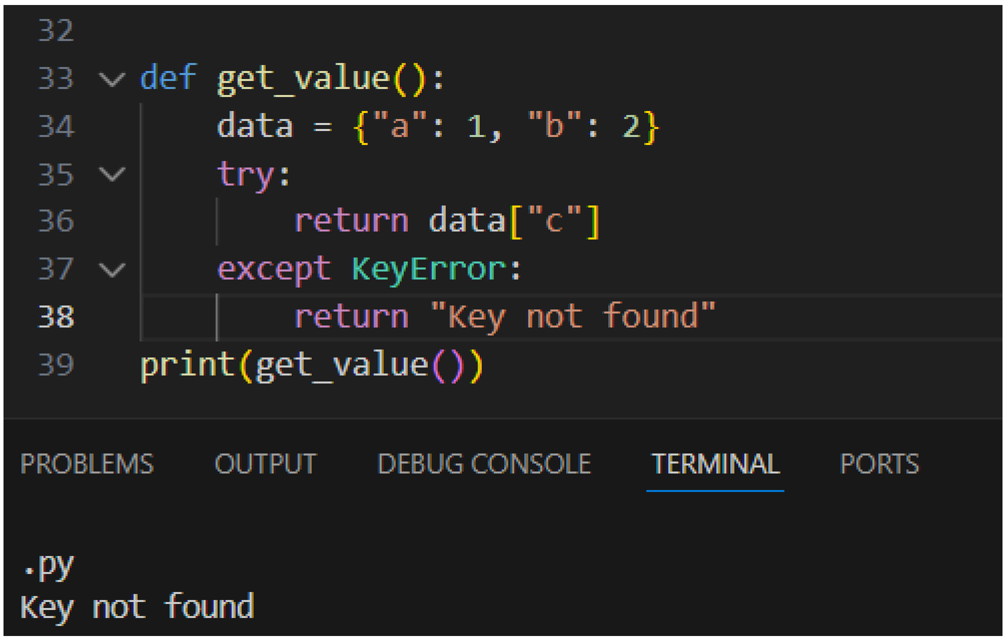
def get_value():

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

return data["c"]

print(get_value())

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


```python
def get_value():
    data = {"a": 1, "b": 2}
    try:
        return data["c"]
    except KeyError:
        return "Key not found"
print(get_value())
```

```
.py
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

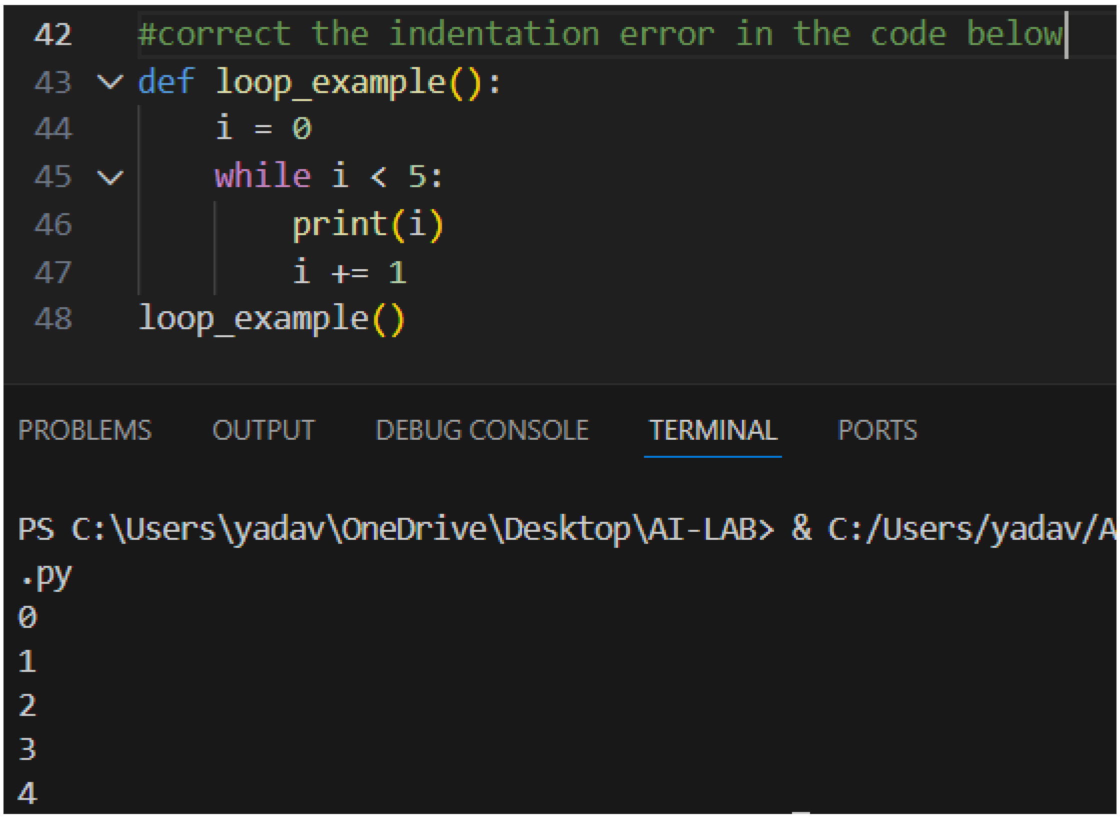
def loop_example():

i = 0

while i < 5:

print(i)

Expected Output: Corrected loop increments



```python
#correct the indentation error in the code below
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
loop_example()
```

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> & C:/Users/yadav/A
.py
0
1
2
3
4
```

Task 6 (Unpacking Error – Wrong Variables)

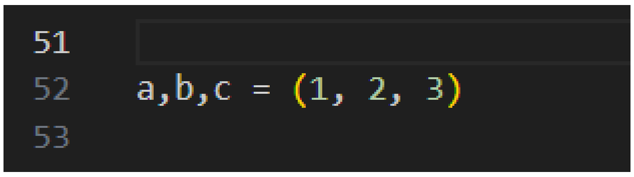
Task: Analyze given code where tuple unpacking fails. Use AI to

fix it.

# Bug: Wrong unpacking

a, b = (1, 2, 3)

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



```python

a,b,c = (1, 2, 3)

```

Task 7 (Mixed Indentation – Tabs vs Spaces)

Task: Analyze given code where mixed indentation breaks

execution. Use AI to fix it.

# Bug: Mixed indentation

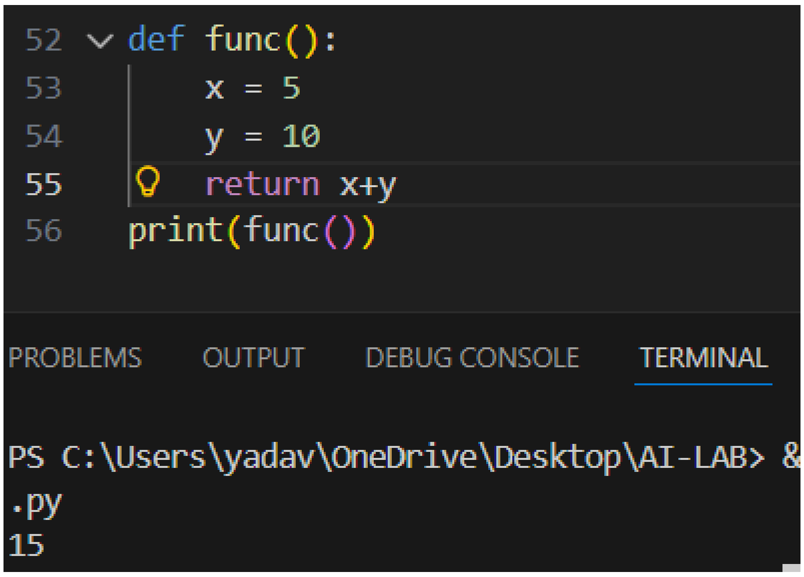
def func():

x = 5

y = 10

return x+y

Expected Output : Consistent indentation applied.



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

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> &
.py
15
```

Task 8 (Import Error – Wrong Module Usage)

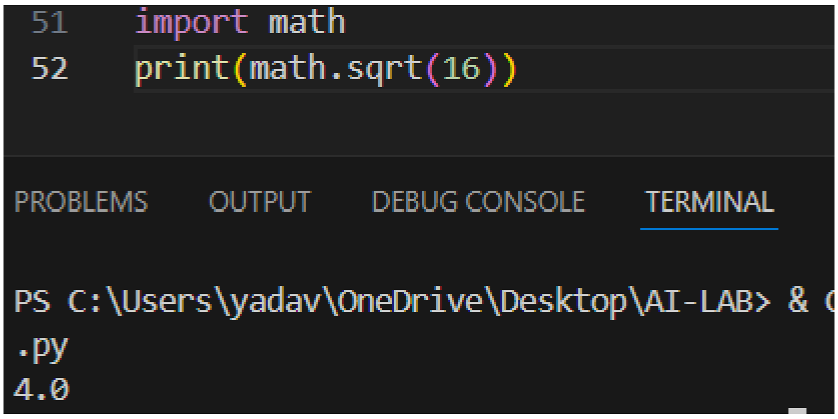
Task: Analyze given code with incorrect import. Use AI to fix.

# Bug: Wrong import

import maths

print(maths.sqrt(16))

Expected Output: Corrected to import math



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

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> &
.py
4.0
```

Task 9 (Unreachable Code – Return Inside Loop)

Task: Analyze given code where a return inside a loop prevents full iteration. Use AI to fix it.

# Bug: Early return inside loop

def total(numbers):

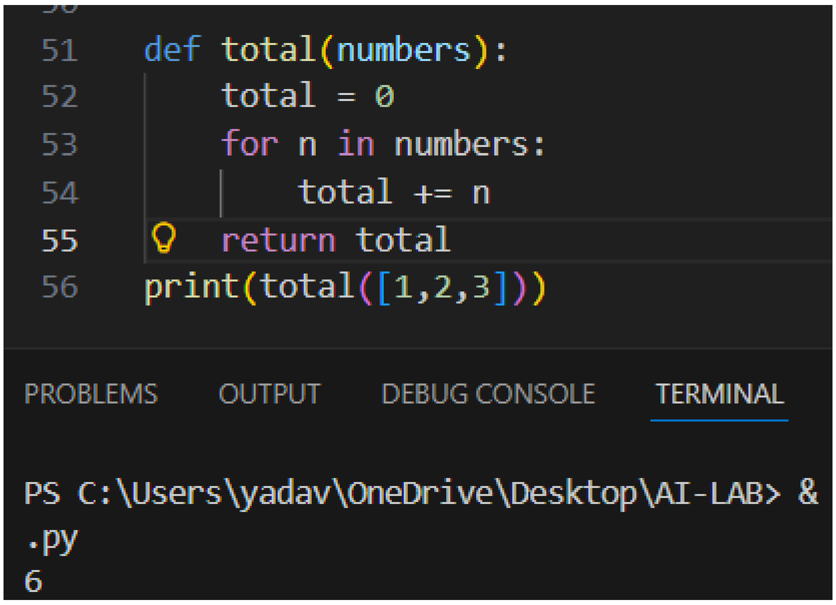
for n in numbers:

return n

print(total([1,2,3]))

Expected Output: Corrected code accumulates sum and returns

after loop.



```
def total(numbers):
    total = 0
    for n in numbers:
        total += n
    return total
print(total([1,2,3]))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> &
.py
6
```

Task 10 (Name Error – Undefined Variable)

Task: Analyze given code where a variable is used before being

defined. Let AI detect and fix the error.

# Bug: Using undefined variable

def calculate_area():

return length * width

print(calculate_area())

Requirements:

• Run the code to observe the error.

• Ask AI to identify the missing variable definition.

• Fix the bug by defining length and width as parameters.

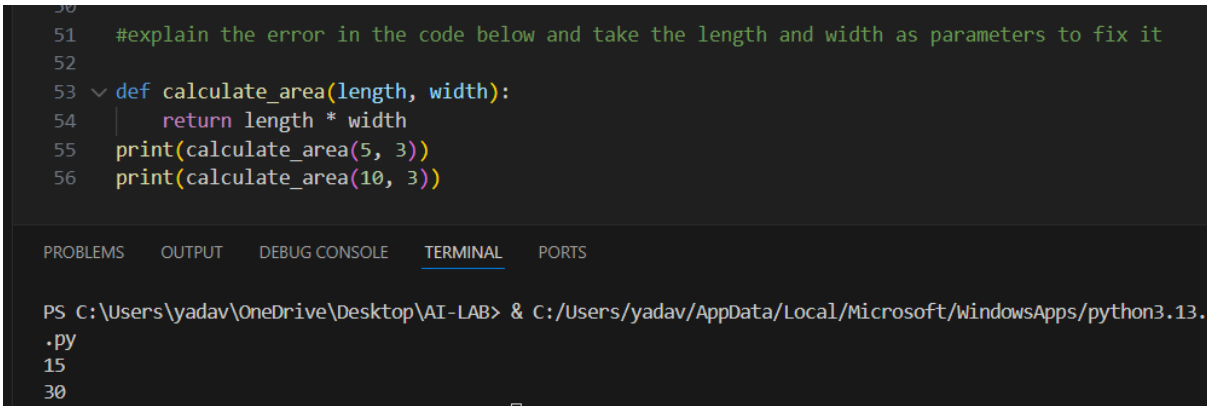
• Add 3 assert test cases for correctness.

Expected Output :

• Corrected code with parameters.

• AI explanation of the bug.

Successful execution of assertions.



```python
#explain the error in the code below and take the length and width as parameters to fix it

def calculate_area(length, width):
    return length * width
print(calculate_area(5, 3))
print(calculate_area(10, 3))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.
.py
15
30
```

Task 11 (Type Error – Mixing Data Types Incorrectly)

Task: Analyze given code where integers and strings are added

incorrectly. Let AI detect and fix the error.

# Bug: Adding integer and string

def add_values():

return 5 + "10"

print(add_values())

Requirements:

• Run the code to observe the error.

• AI should explain why int + str is invalid.

• Fix the code by type conversion (e.g., int("10") or str(5)).

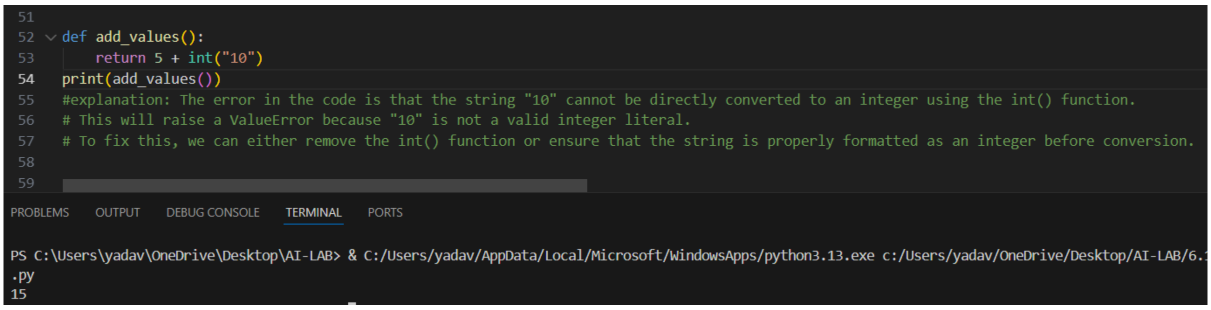
• Verify with 3 assert cases.

Expected Output #6:

• Corrected code with type handling.

• AI explanation of the fix.

Successful test validation.



```
def add_values():
    return 5 + int("10")
print(add_values())
#explanation: The error in the code is that the string "10" cannot be directly converted to an integer using the int() function.
# This will raise a ValueError because "10" is not a valid integer literal.
# To fix this, we can either remove the int() function or ensure that the string is properly formatted as an integer before conversion.
```

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/yadav/OneDrive/Desktop/AI-LAB/6.
.py
15
```

Task 12 (Type Error – String + List Concatenation)

Task: Analyze code where a string is incorrectly added to a list.

# Bug: Adding string and list

def combine():

return "Numbers: " + [1, 2, 3]

print(combine())

Requirements:

• Run the code to observe the error.

• Explain why str + list is invalid.

• Fix using conversion (str([1,2,3]) or " ".join()).

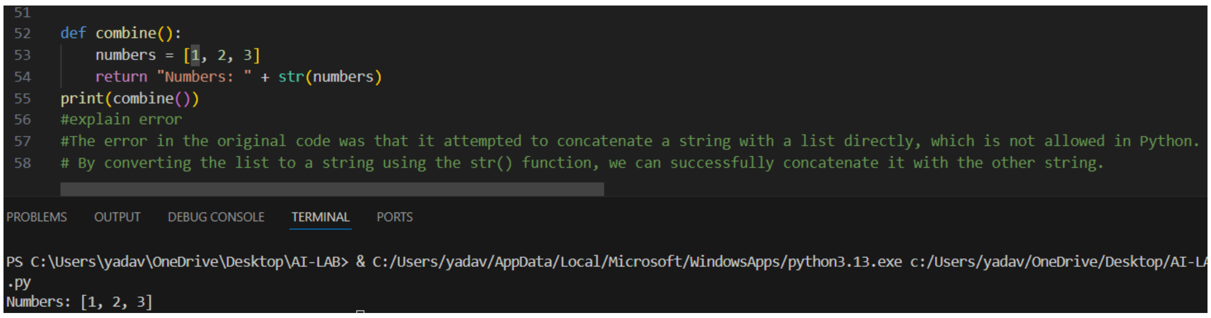
• Verify with 3 assert cases.

Expected Output:

• Corrected code

• Explanation

• Successful test validation



```
def combine():
    numbers = [1, 2, 3]
    return "Numbers: " + str(numbers)
print(combine())
#explain error
#The error in the original code was that it attempted to concatenate a string with a list directly, which is not allowed in Python.
# By converting the list to a string using the str() function, we can successfully concatenate it with the other string.
```

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/yadav/OneDrive/Desktop/AI-L
.py
Numbers: [1, 2, 3]
```

Task 13 (Type Error – Multiplying String by Float)

Task: Detect and fix code where a string is multiplied by a float.

# Bug: Multiplying string by float

def repeat_text():

return "Hello" * 2.5

print(repeat_text())

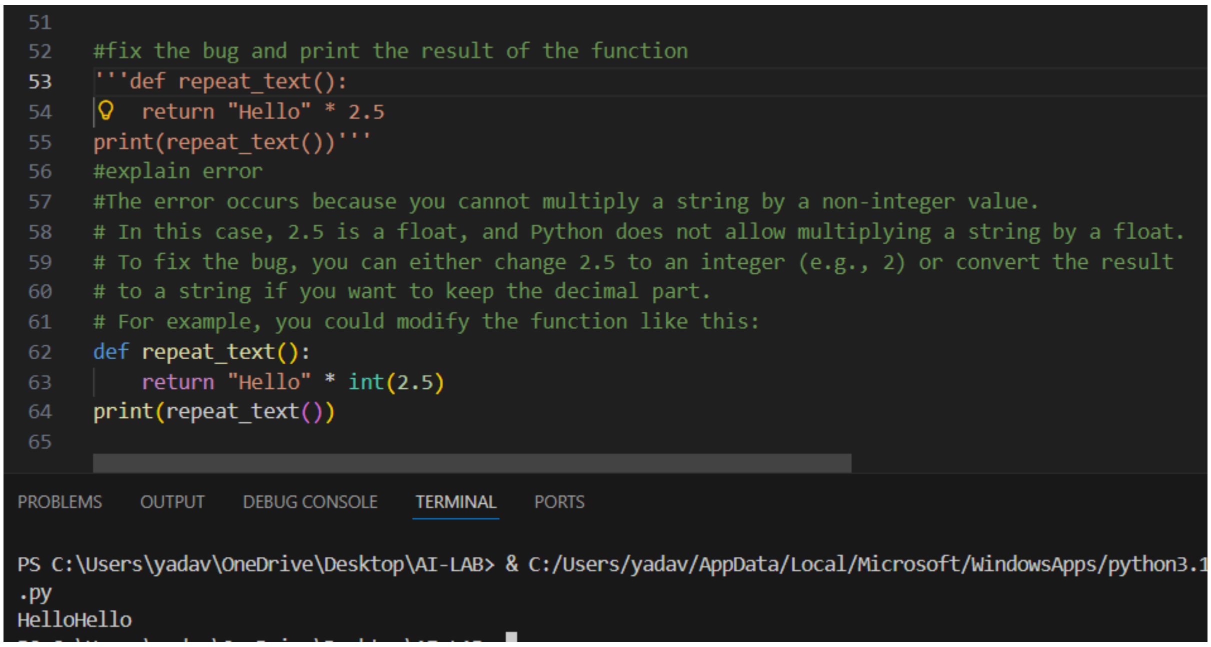
Requirements:

• Observe the error.

• Explain why float multiplication is invalid for strings.

• Fix by converting float to int.

• Add 3 assert test cases.



```
#fix the bug and print the result of the function
'''def repeat_text():
    return "Hello" * 2.5
print(repeat_text())'''
#explain error
#The error occurs because you cannot multiply a string by a non-integer value.
# In this case, 2.5 is a float, and Python does not allow multiplying a string by a float.
# To fix the bug, you can either change 2.5 to an integer (e.g., 2) or convert the result
# to a string if you want to keep the decimal part.
# For example, you could modify the function like this:
def repeat_text():
    return "Hello" * int(2.5)
print(repeat_text())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.1
.py
HelloHello
```

Task 14 (Type Error – Adding None to Integer)

Task: Analyze code where None is added to an integer.

# Bug: Adding None and integer

def compute():

value = None

return value + 10

print(compute())

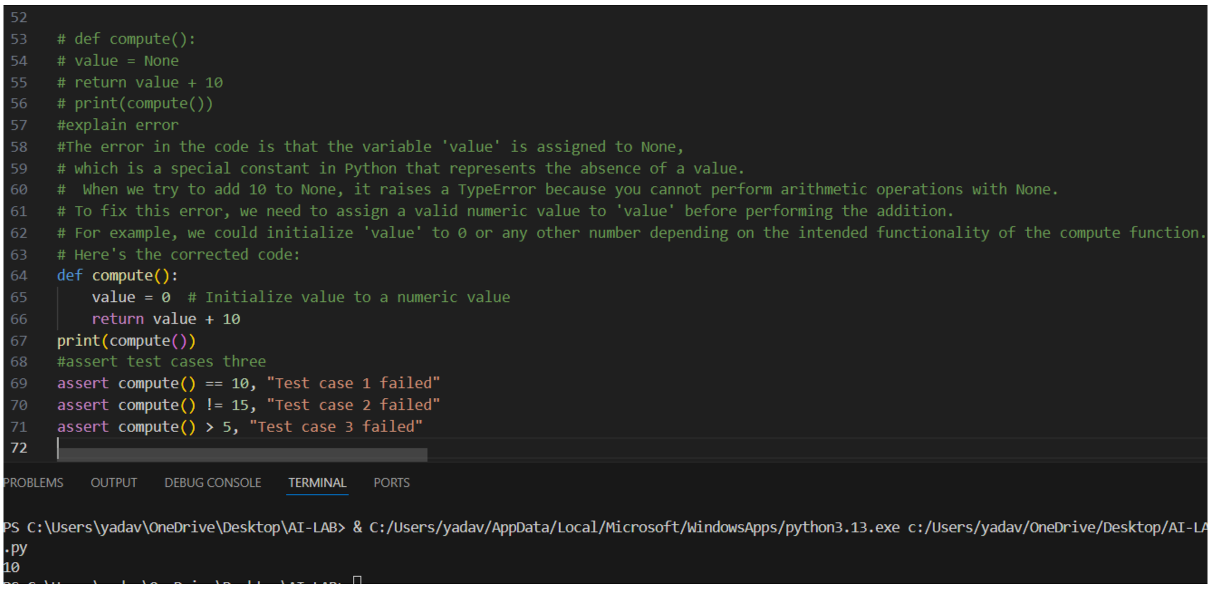
Requirements:

• Run and identify the error.

• Explain why NoneType cannot be added.

• Fix by assigning a default value.

• Validate using asserts.



```
# def compute():
# value = None
# return value + 10
# print(compute())
#explain error
#The error in the code is that the variable 'value' is assigned to None,
# which is a special constant in Python that represents the absence of a value.
#  When we try to add 10 to None, it raises a TypeError because you cannot perform arithmetic operations with None.
# To fix this error, we need to assign a valid numeric value to 'value' before performing the addition.
# For example, we could initialize 'value' to 0 or any other number depending on the intended functionality of the compute function.
# Here's the corrected code:
def compute():
    value = 0  # Initialize value to a numeric value
    return value + 10
print(compute())
#assert test cases three
assert compute() == 10, "Test case 1 failed"
assert compute() != 15, "Test case 2 failed"
assert compute() > 5, "Test case 3 failed"
```

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/yadav/OneDrive/Desktop/AI-LA
.py
10
```

Task 15 (Type Error – Input Treated as String Instead of

Number)

Task: Fix code where user input is not converted properly.

# Bug: Input remains string

def sum_two_numbers():

a = input("Enter first number: ")

b = input("Enter second number: ")

return a + b

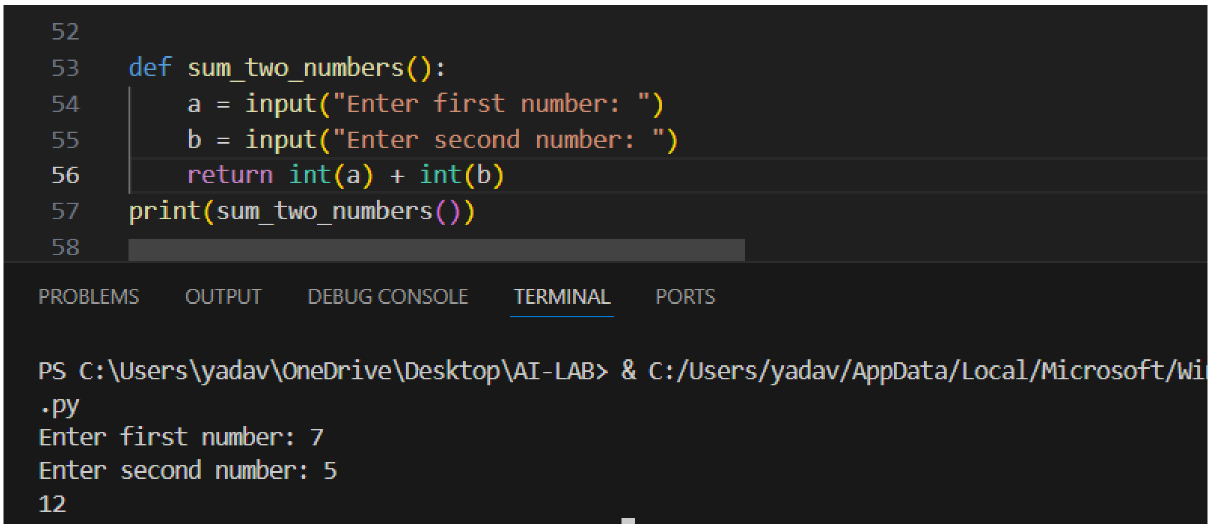
print(sum_two_numbers())

Requirements:

• Explain why input is always string.

• Fix using int() conversion.

• Verify with assert test cases



```python
def sum_two_numbers():
    a = input("Enter first number: ")
    b = input("Enter second number: ")
    return int(a) + int(b)
print(sum_two_numbers())
```

```
PS C:\Users\yadav\OneDrive\Desktop\AI-LAB> & C:/Users/yadav/AppData/Local/Microsoft/Wi
.py
Enter first number: 7
Enter second number: 5
12
```