AI Assisted Coding

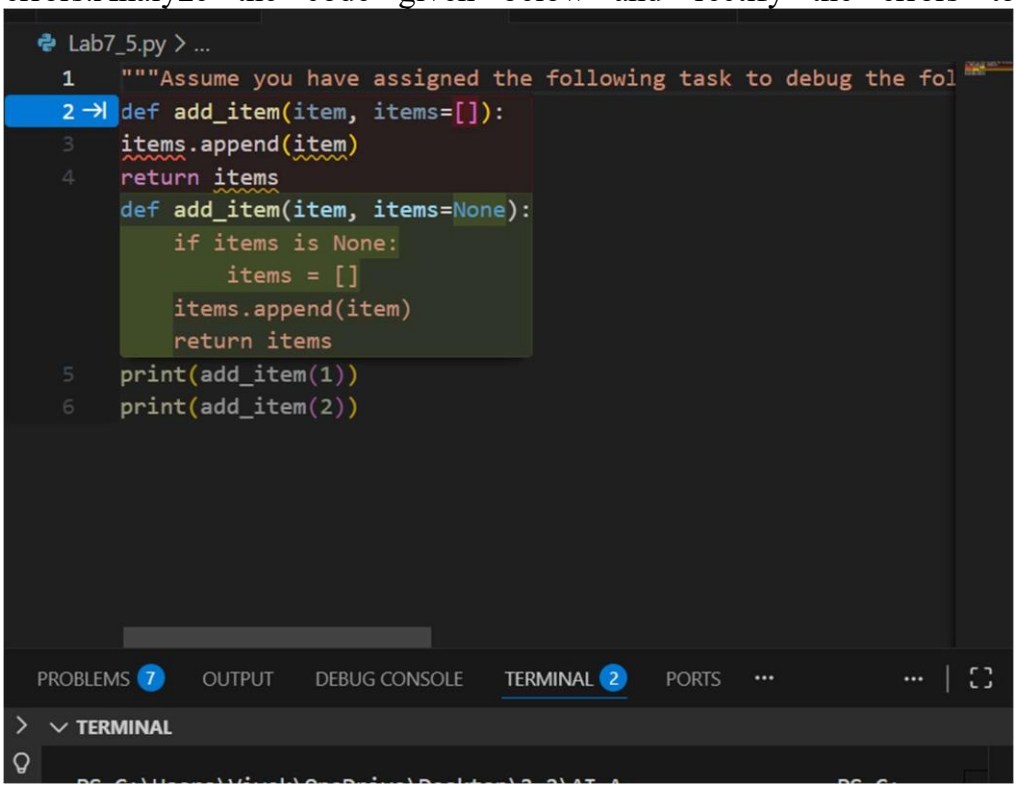
Assignment 7.5

Name: Dinesh

Hallticket:2303A52329

Task_01:
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.

Prompt:
"""Assume you have assigned the following task to debug the following code which has some errors.Analyze the code given below and rectify the errors to get the output"""



Task-02
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

```
def check_sum():
    return (0.1 + 0.2) == 0.3
print(check_sum())
```

Output :

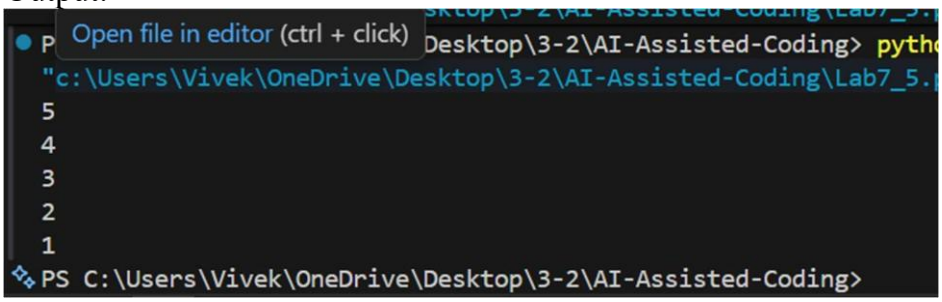
```
False
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Codi
"c:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding
False
```

Task-03:
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.
Prompt:

```
""" debug this code and re-write the correct code below"""
# Bug: No base case
def countdown(n):
print(n)
return countdown(n-1)
countdown(5)
```

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

Output:



Task_04:
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.

```
# Bug: Accessing non-existing key
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())
```

Output:

```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u
5.py"
Key not found
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

Task_05:
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.
```

Prompt:
""" You are a programming assistant. Analyze the following Python code and identify why the loop never ends. fix the code so that output prints numbers from 0 to 4."""

```
""" You are a programming assistant. Analyze the following Python code and identify why the loop never ends. fix the
def loop_example():
i = 0
while i < 5:
print(i)
    i += 1
loop_example()
# Bug: Infinite loop due to missing increment of i
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
    i = 0
    while i < 5:
        print(i)
        i += 1

loop_example()
```

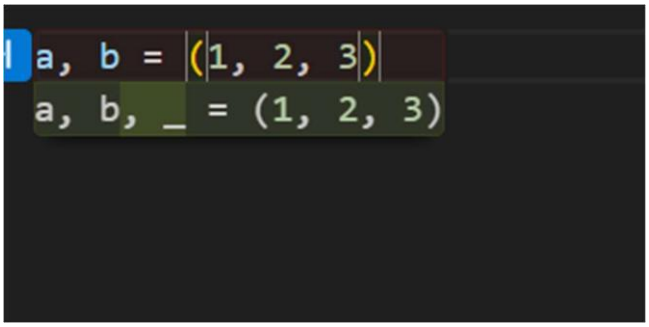
Keep Undo

Output:

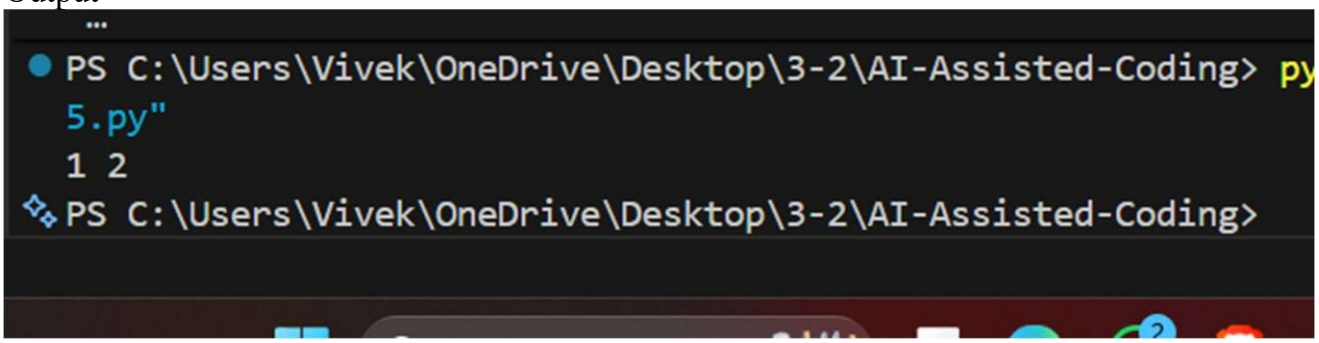
```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u "c:\Us
5.py"
0
1
2
3
4
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

Task_06:
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.

Prompt:
""" debug this code and re-write the correct code below by Correct unpacking or using _ for extra values.”””””



Output



"""The error occurs because there are three values in the tuple (1, 2, 3) but only two variables (a and b) to unpack them into. To fix this, we can use an underscore (_) as a placeholder for the extra value that we don't need. This way, we can unpack the first two values into a and b, and ignore the third value."""

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.

Prompt: """" debug this code by giving proper indentation and re-write the correct code below give inline suggestions where necessary""""

Code:

```
debug this code by giving proper indentation and re-write the correct c
def func():
x = 5
y = 10
return x+y
print(func())
def func():
    x = 5          # Indent the line to be inside the function
    y = 10         # Indent the line to be inside the function
    return x + y   # Indent the line to be inside the function
```

Output:

```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> py
5.py"
15
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> 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
Prompt:
""" debug this code by taking correct import module and re-write the correct code"""
Code:

```
47  """ debug this code by taking correct import module and re-w
48  import maths
49  print(maths.sqrt(16))   print(math.sqrt(16))
50
```

Output:

```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> pyth
5.py"
4.0
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

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.

Prompt: """ debug this code by Analyze given code where a return inside a loop prevents full iteration. """

Code:

```
46  # print(func())        # This line is corr
47  # """ debug this code by taking correct
48  # import math
49  # print(math.sqrt(16))
50  """ debug this code by Analyze given cod
51  def total(numbers):
52      total = 0
53      for n in numbers:
54          return n
    print(total([1,2,3]))
```

Output:

```
ssisted-Coding> python -u "c:\Users\Vivek
neDrive\Desktop\3-2\AI-Assisted-Coding\La
_5.py"
6
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI
ssisted-Coding>
```

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 :

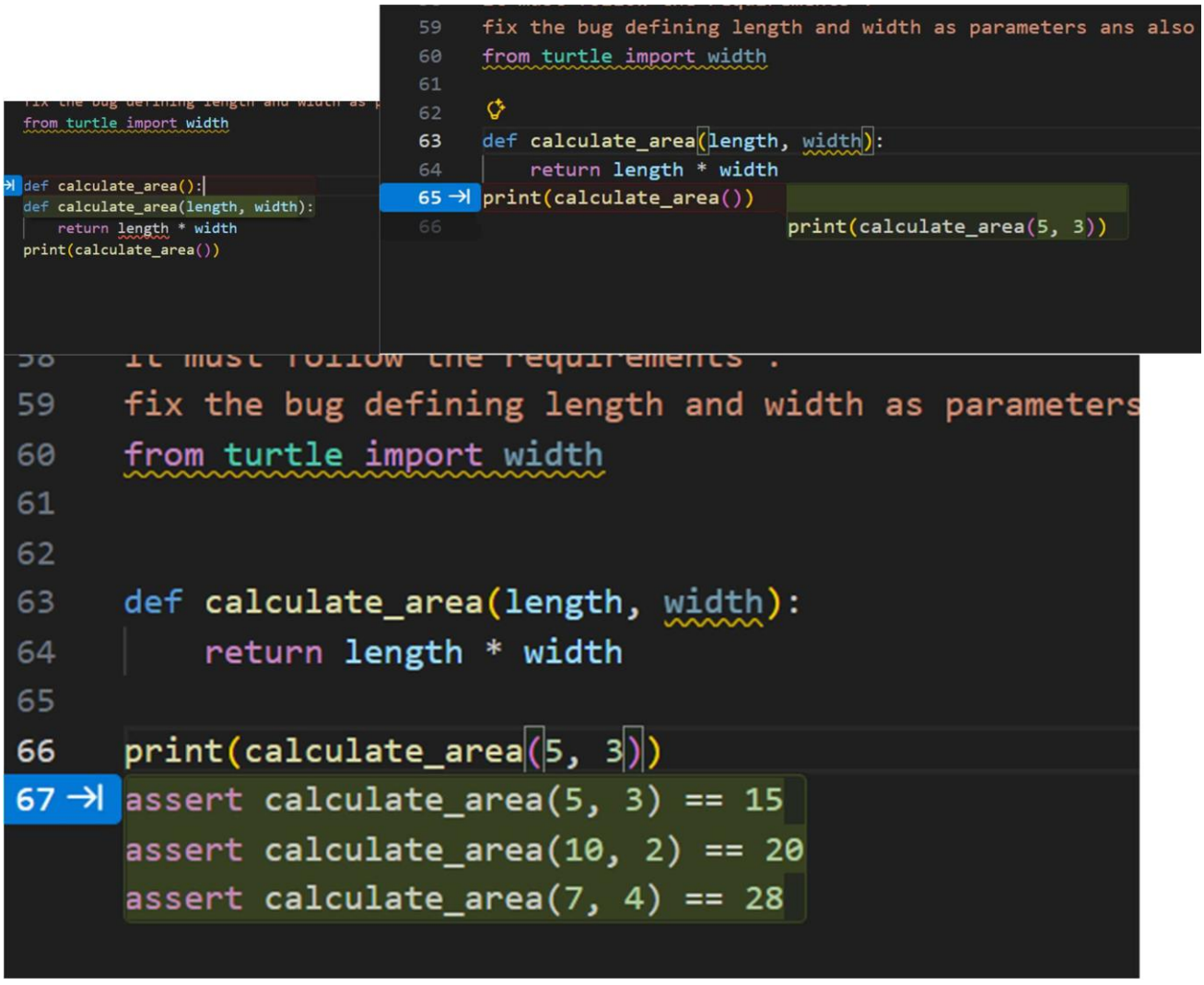
- Corrected code with parameters.
- AI explanation of the bug.

Successful execution of assertions.

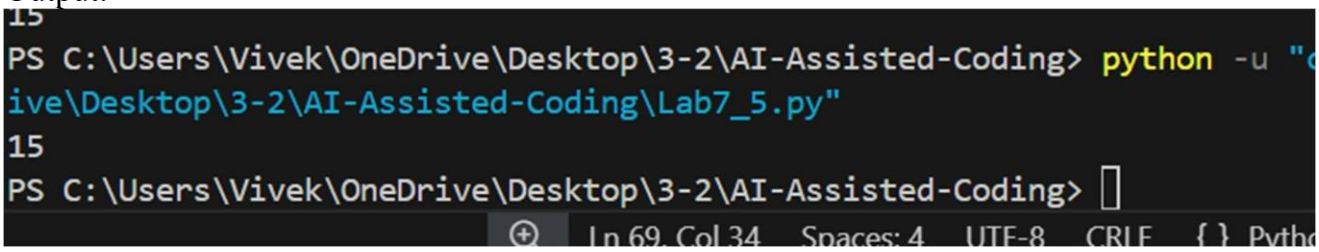
Prompt:

debug the given below code by Analyze given code where a variable is used before being defined.
it must follow the requirements :
fix the bug defining length and width as parameters ans also add 3 asserts test cases"""

Code:



Output:



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:

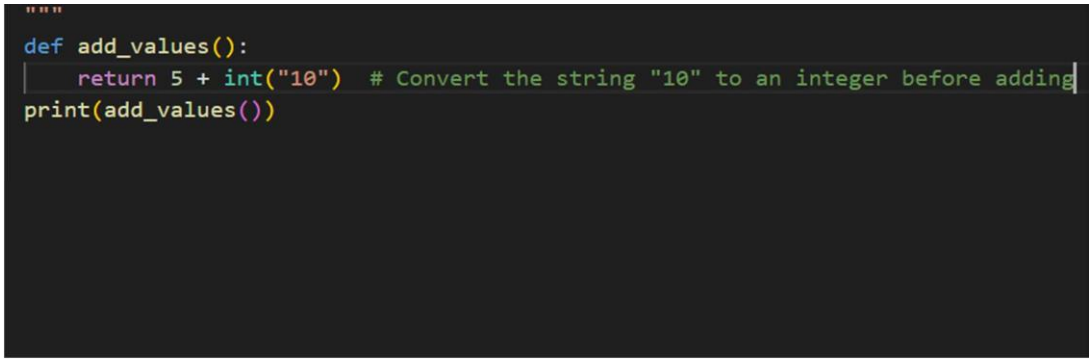
- Corrected code with type handling.
- AI explanation of the fix.

Successful test validation.

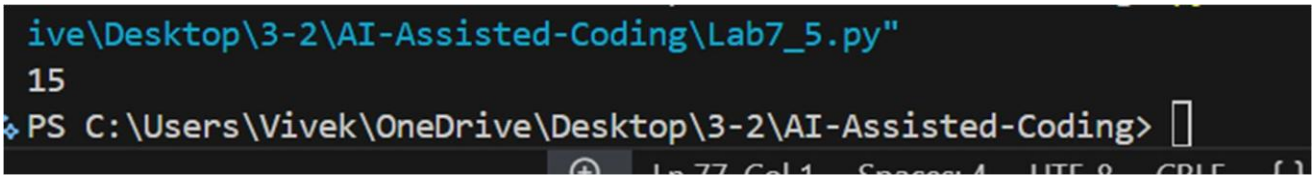
Prompt:

""" debug the given code by Analyze given code where Analyze given code where integers and strings are added incorrectly.I need this requirements in the code • Fix the code by type conversion Verify with 3 assert cases give in line suggestions where necessary"""

Code:



Output:



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:

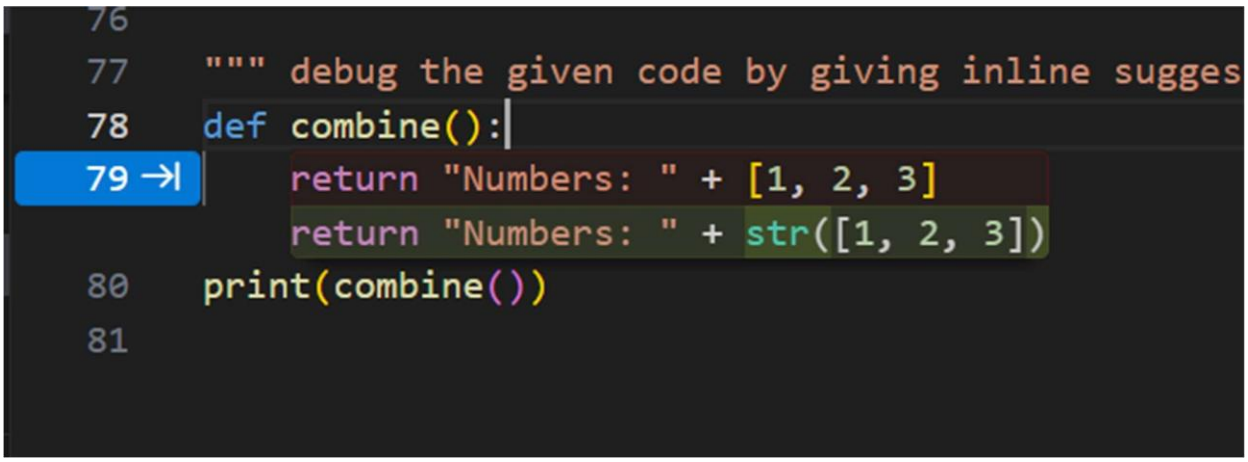
- Run the code to observe the error.
- Explain why str + list is invalid.
- Fix using conversion (str([1,2,3]) or " ".join()).
- Verify with 3 assert cases.

Expected Output:

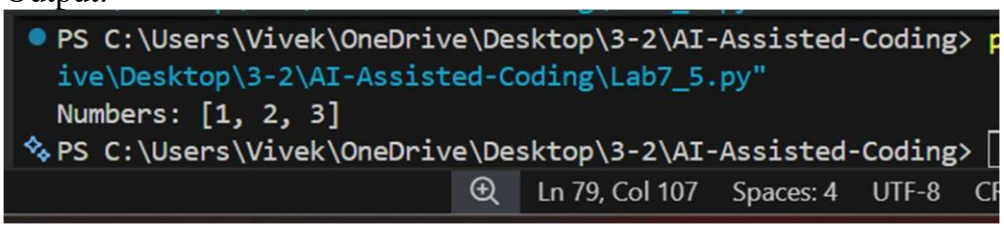
- Corrected code
- Explanation
- Successful test validation

Prompt: """ debug the given code by giving inline suggestions where necessary and Analyze given code where Analyze given code where a list is concatenated with a string without proper conversion. """

Code:



Output:

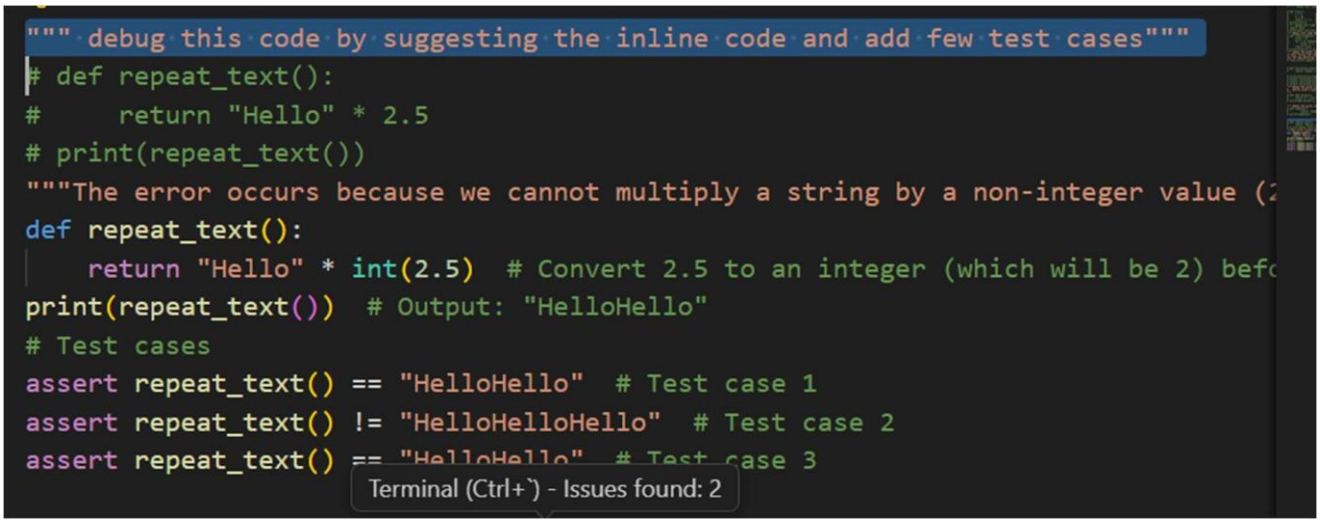


Task 13 (Type Error – Multiplying String by Float)
Task: Detect and fix code where a string is multiplied by a float.
# Bug: Multiplying string by float
def repeat_text():
    return "Hello" * 2.5
print(repeat_text())
Requirements:

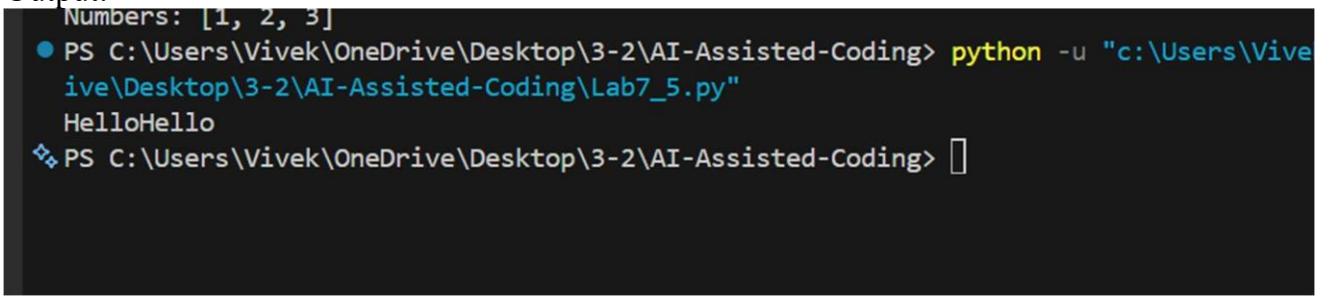
- Observe the error.
- Explain why float multiplication is invalid for strings.
- Fix by converting float to int.
- Add 3 assert test cases.

Prompt:
""" debug this code by suggesting the inline code and add few test cases"""

Code:



Output:



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.

Prompt:
""" debug this code by suggesting the inline code and add few test cases requirements: • Run and identify the error.
Fix by assigning a default value. Validate using asserts.
"""
Code:

```
98      • Validate using asserts.
99  """
100 def compute():
101     value = None
        if value is None:
            value = 0  # Assign a default value of 0
102     return value + 10
103
104 print(compute())
105
```

Output:

```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding
ive\Desktop\3-2\AI-Assisted-Coding\Lab7_5.py"
10
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding
```

Ln 105, Col 17 Spaces: 4 UTF-8

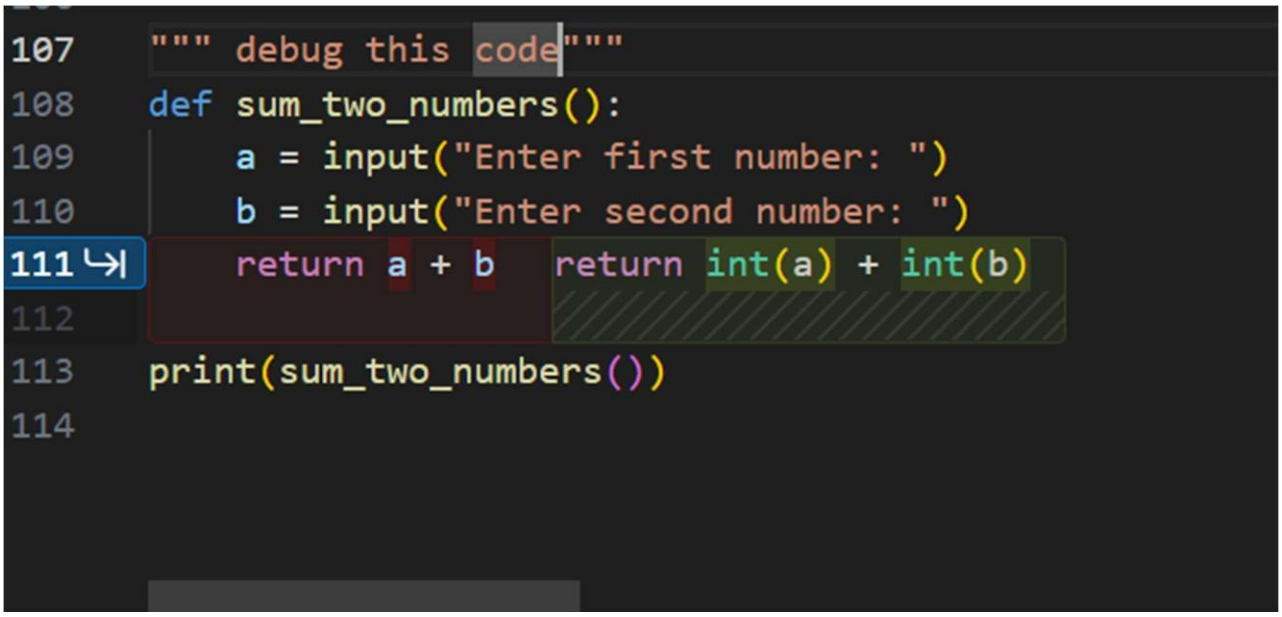
Task 15 (Type Error – Input Treated as String Instead of Number)
Task: Fix code where user input is not converted properly.
# Bug: Input remains string
def sum_two_numbers():
    a = input("Enter first number: ")
    b = input("Enter second number: ")
    return a + b

print(sum_two_numbers())
Requirements:

- Explain why input is always string.
- Fix using int() conversion.

Verify with assert test cases

Code:



Output: