Course Title	AI Assisted Coding

Name: G. Snehith Roll Number: 2303A52324 Batch: 45

Assignment Number: 7.5(Present assignment number)/**24**(Total number of assignments)

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.

Refactor buggy code using responsible and reliable programming patterns.

Task 1 (Mutable Default Argument – Function Bug)

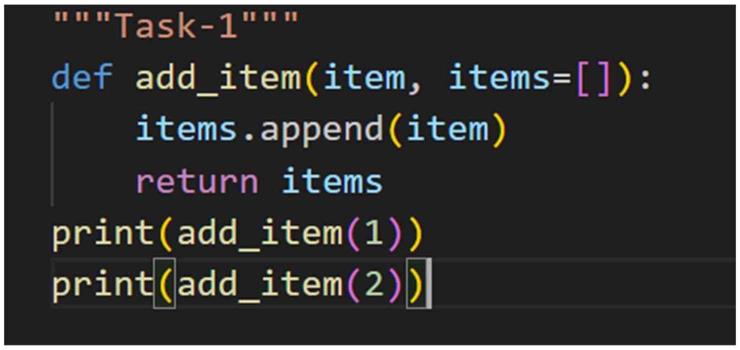
Task: Analyze given code where a mutable default argument causes unexpected behavior. Use AI to fix it.

```
# 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.

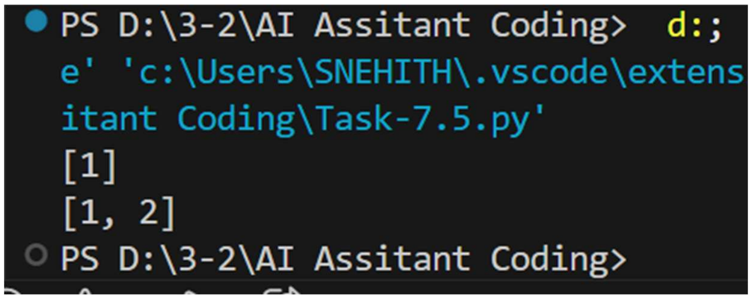
Prompt: Solve the Error

Code:



```
"""Task-1"""
def add_item(item, items=[]):
    items.append(item)
    return items
print(add_item(1))
print(add_item(2))
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:;
e' 'c:\Users\SNEHITH\.vscode\extens
itant Coding\Task-7.5.py'
[1]
[1, 2]
PS D:\3-2\AI Assitant Coding>
```

Task 2 (Floating-Point Precision Error)

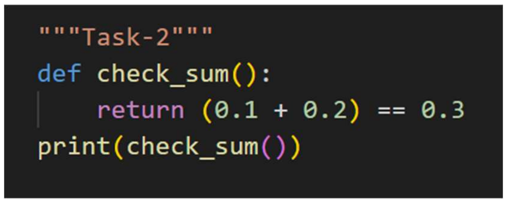
Task: Analyze given code where floating-point comparison fails. Use AI to correct with tolerance.

```
# Bug: Floating point precision issue
def check_sum():
    return (0.1 + 0.2) == 0.3
print(check_sum())
```

Expected Output: Corrected function

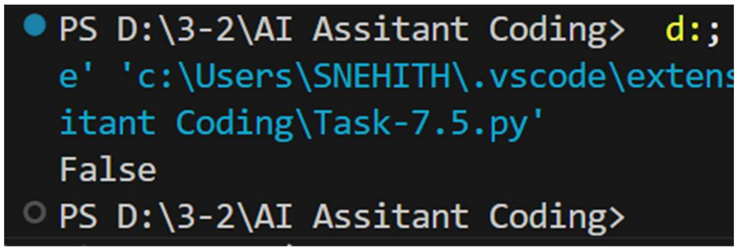
Prompt: Solve this Error

Code:



```
"""Task-2"""
def check_sum():
    return (0.1 + 0.2) == 0.3
print(check_sum())
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:;
e' 'c:\Users\SNEHITH\.vscode\extens
itant Coding\Task-7.5.py'
False
PS D:\3-2\AI Assitant Coding>
```

Task 3 (Recursion Error – Missing Base Case)

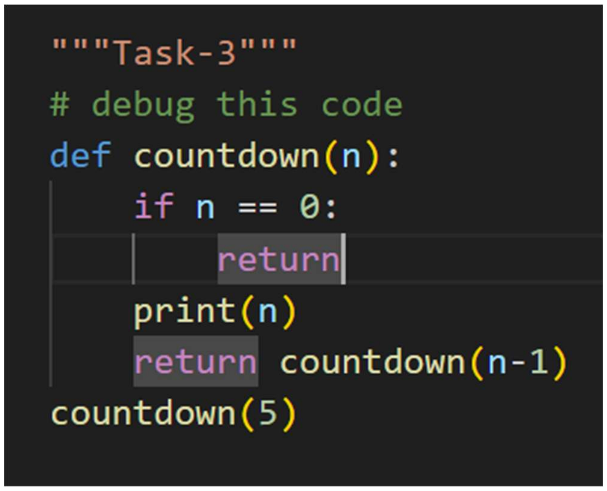
Task: Analyze given code where recursion runs infinitely due to missing base case. Use AI to fix.

```
# Bug: No base case
def countdown(n):
    print(n)
    return countdown(n-1)
countdown(5)
```

Expected Output : Correct recursion with stopping condition**.**

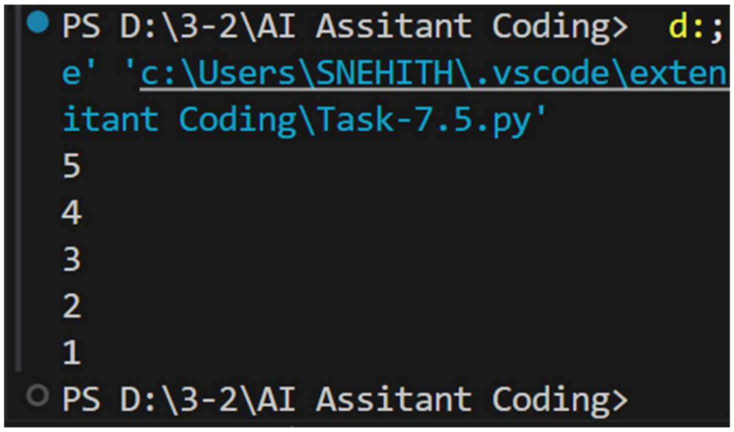
Prompt: Solve this Error

Code:



```
"""Task-3"""
# debug this code
def countdown(n):
    if n == 0:
        return
    print(n)
    return countdown(n-1)
countdown(5)
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:;
e' 'c:\Users\SNEHITH\.vscode\exten
itant Coding\Task-7.5.py'
5
4
3
2
1
PS D:\3-2\AI Assitant Coding>
```

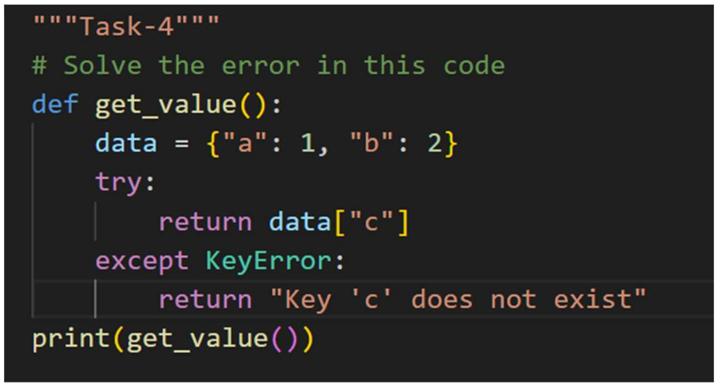
Task 4 (Dictionary Key Error)
Task: Analyze given code where a missing dictionary key causes error. Use AI to fix it.

```
# 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.

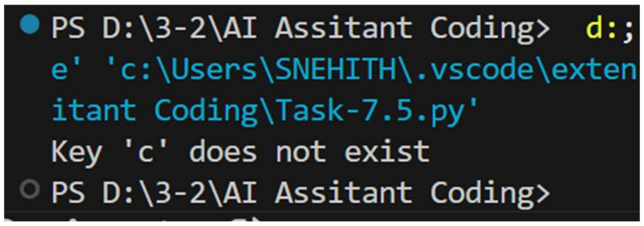
Prompt: Solve this error of Accessing non-existing key

Code:



```python
"""Task-4"""
# Solve the error in this code
def get_value():
    data = {"a": 1, "b": 2}
    try:
        return data["c"]
    except KeyError:
        return "Key 'c' does not exist"
print(get_value())
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:;
e' 'c:\Users\SNEHITH\.vscode\exten
itant Coding\Task-7.5.py'
Key 'c' does not exist
PS D:\3-2\AI Assitant Coding>
```

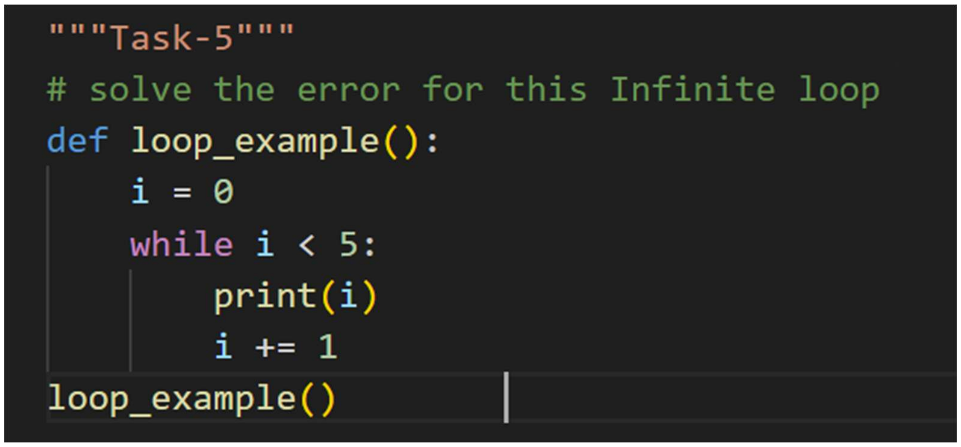
Task 5 (Infinite Loop – Wrong Condition)
Task: Analyze given code where loop never ends. Use AI to detect and fix it.

```
# Bug: Infinite loop
def loop_example():
    i = 0
    while i < 5:
        print(i)
```

Expected Output: Corrected loop increments i.

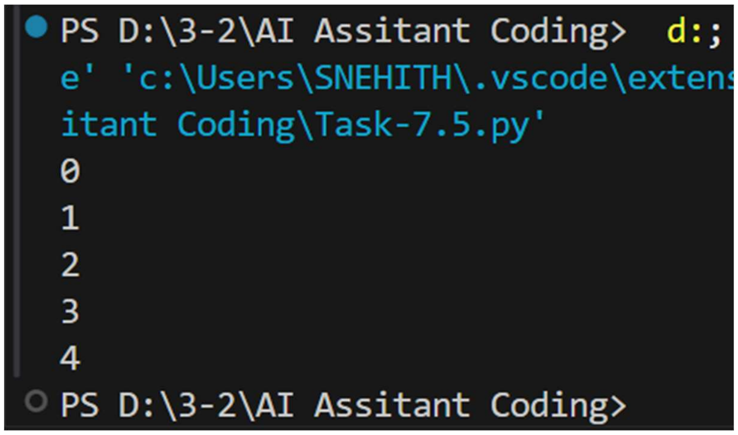
Prompt: # solve the error for this Infinite loop

Code:



```
"""Task-5"""
# solve the error for this Infinite loop
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
loop_example()
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:;
e' 'c:\Users\SNEHITH\.vscode\extens
itant Coding\Task-7.5.py'
0
1
2
3
4
PS D:\3-2\AI Assitant Coding>
```

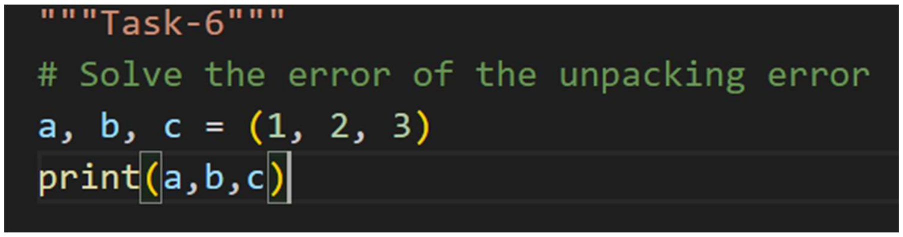
Task 6 (Unpacking Error – Wrong Variables)
Task: Analyze given code where tuple unpacking fails. Use AI to fix it.

```
# Bug: Wrong unpacking
a, b = (1, 2, 3)
```

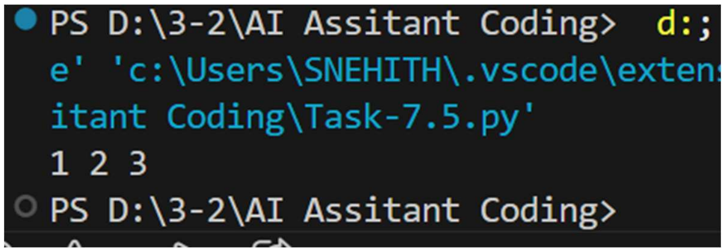
Expected Output: Correct unpacking or using _ for extra values.

Prompt: # Solve the error of the unpacking error

Code:


```
"""Task-6"""
# Solve the error of the unpacking error
a, b, c = (1, 2, 3)
print(a,b,c)
```

Output:


```
PS D:\3-2\AI Assitant Coding>  d:;
e' 'c:\Users\SNEHITH\.vscode\exten
itant Coding\Task-7.5.py'
1 2 3
PS D:\3-2\AI Assitant Coding>
```

Task 7 (Mixed Indentation – Tabs vs Spaces)

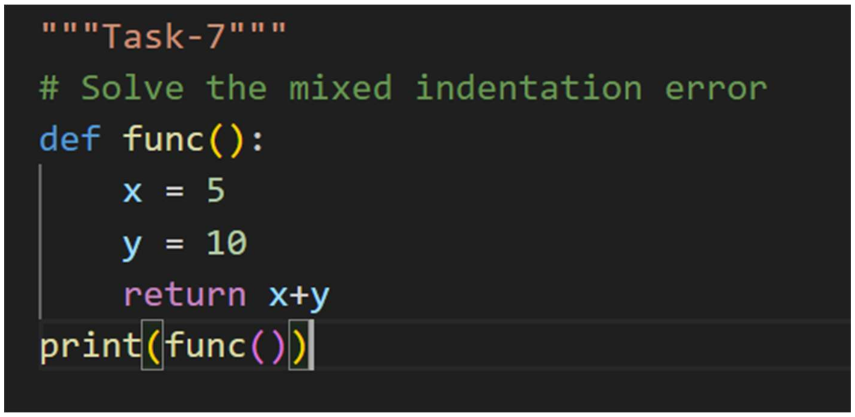
Task: Analyze given code where mixed indentation breaks execution. Use AI to fix it.

```
# Bug: Mixed indentation
def func():
    x = 5
        y = 10
    return x+y
```

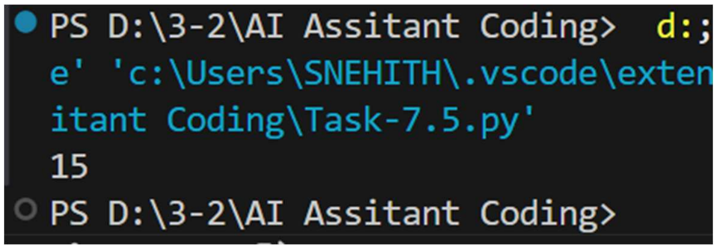
Expected Output : Consistent indentation applied

Prompt: # Solve the mixed indentation error

Code:


```
"""Task-7"""
# Solve the mixed indentation error
def func():
    x = 5
    y = 10
    return x+y
print(func())
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:;
e' 'c:\Users\SNEHITH\.vscode\exten
itant Coding\Task-7.5.py'
15
PS D:\3-2\AI Assitant Coding>
```

Task 8 (Import Error – Wrong Module Usage)

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

# Bug: Wrong import

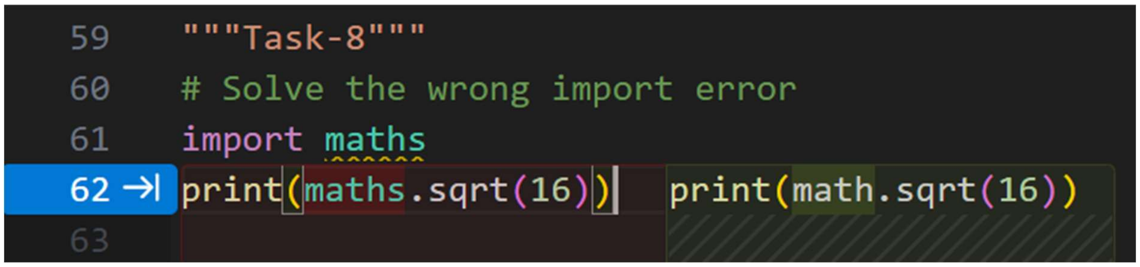
import maths

print(maths.sqrt(16))

Expected Output: Corrected to import math

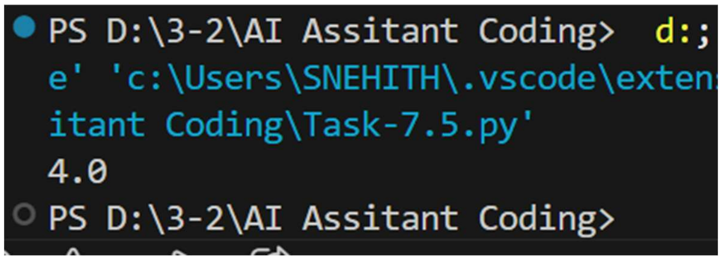
Prompt: # Solve the wrong import error

Code:



```
"""Task-8"""
# Solve the wrong import error
import maths
print(maths.sqrt(16))    print(math.sqrt(16))
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:;
e' 'c:\Users\SNEHITH\.vscode\exten
itant Coding\Task-7.5.py'
4.0
PS D:\3-2\AI Assitant 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.

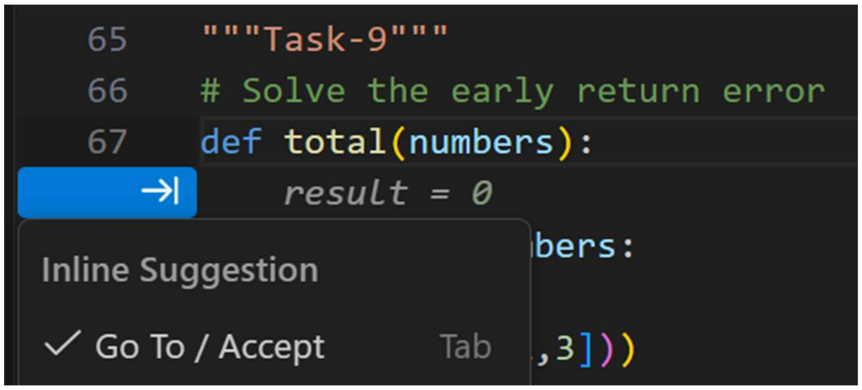
```
# Bug: Early return inside loop
def total(numbers):
    for n in numbers:
        return n
```

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

Expected Output: Corrected code accumulates sum and returns after loop.

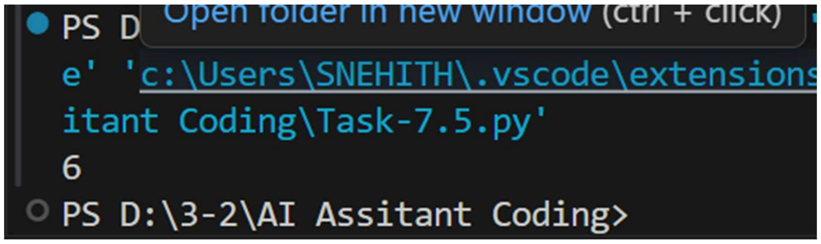
Prompt:

Code:



```
"""Task-9"""
# Solve the early return error
def total(numbers):
    result = 0
                        bers:

                        ,3]))
```

Output:



```
PS D  Open folder in new window (ctrl + click)
e' 'c:\Users\SNEHITH\.vscode\extensions
itant Coding\Task-7.5.py'
6
PS D:\3-2\AI Assitant 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.

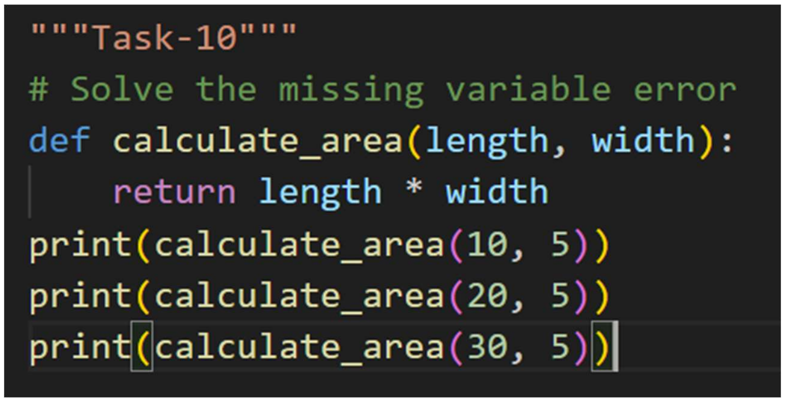
Expected Output :

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

Successful execution of assertions.

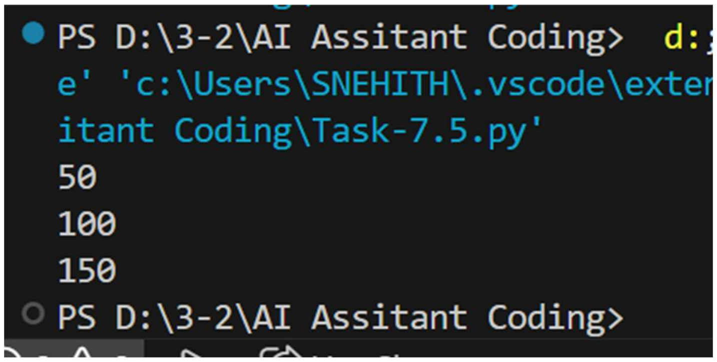
Prompt: # Solve the missing variable error

Code:



```python
"""Task-10"""
# Solve the missing variable error
def calculate_area(length, width):
    return length * width
print(calculate_area(10, 5))
print(calculate_area(20, 5))
print(calculate_area(30, 5))
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:
e'  'c:\Users\SNEHITH\.vscode\exter
itant Coding\Task-7.5.py'
50
100
150
PS D:\3-2\AI Assitant Coding>
```

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.

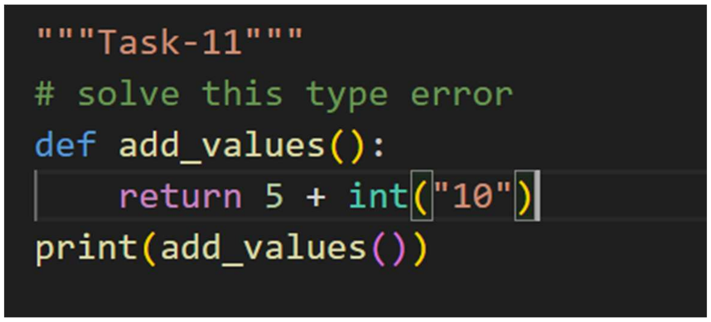
Expected Output #6:

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

Successful test validation.

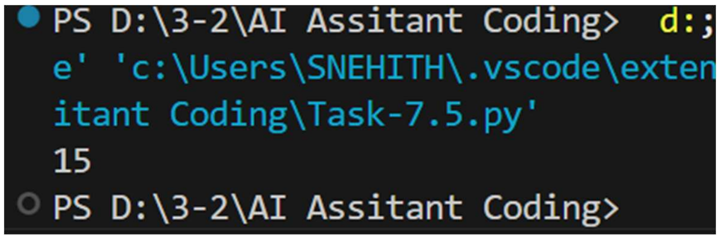
Prompt: # solve this type error

Code:



```python
"""Task-11"""
# solve this type error
def add_values():
    return 5 + int("10")
print(add_values())
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:;
e' 'c:\Users\SNEHITH\.vscode\exten
itant Coding\Task-7.5.py'
15
PS D:\3-2\AI Assitant Coding>
```

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:

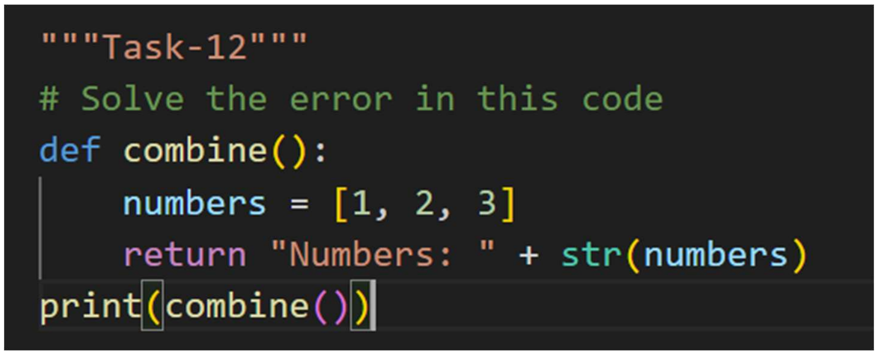
- Run the code to observe the error.
- Explain why str + list is invalid.
- Fix using conversion (str([1,2,3]) or " ".join()).
- Verify with 3 assert cases.

Expected Output:

- Corrected code
- Explanation
- Successful test validation

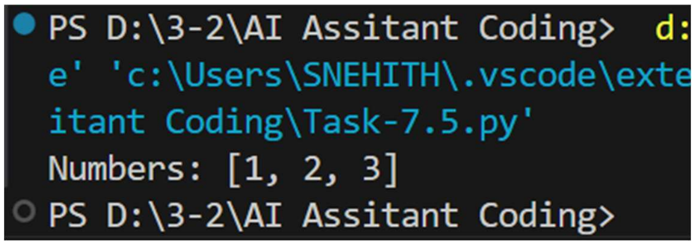
Prompt: # Solve the error in this code

Code:



```python
"""Task-12"""
# Solve the error in this code
def combine():
    numbers = [1, 2, 3]
    return "Numbers: " + str(numbers)
print(combine())
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:
e' 'c:\Users\SNEHITH\.vscode\exte
itant Coding\Task-7.5.py'
Numbers: [1, 2, 3]
PS D:\3-2\AI Assitant Coding>
```

Task 13 (Type Error – Multiplying String by Float)

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

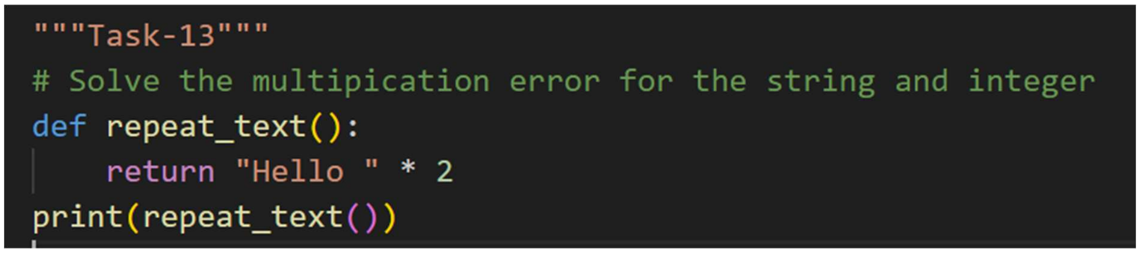
```
# Bug: Multiplying string by float
def repeat_text():
    return "Hello" * 2.5
print(repeat_text())
```

Requirements:

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

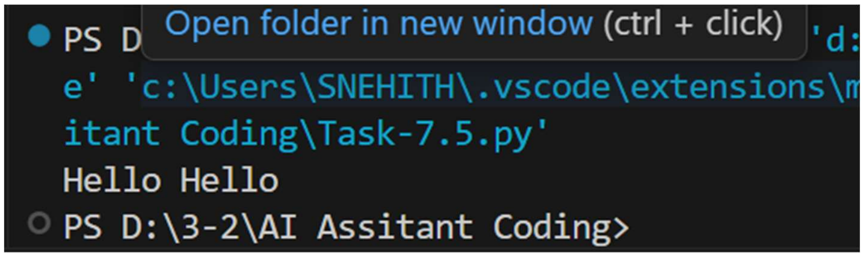
Prompt: # Solve the Multiplication error for the string and integer

Code:



```python
"""Task-13"""
# Solve the multipication error for the string and integer
def repeat_text():
    return "Hello " * 2
print(repeat_text())
```

Output:



```
● PS D  Open folder in new window (ctrl + click)  'd:
  e' 'c:\Users\SNEHITH\.vscode\extensions\m
  itant Coding\Task-7.5.py'
  Hello Hello
○ PS D:\3-2\AI Assitant Coding>
```

Task 14 (Type Error – Adding None to Integer)

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

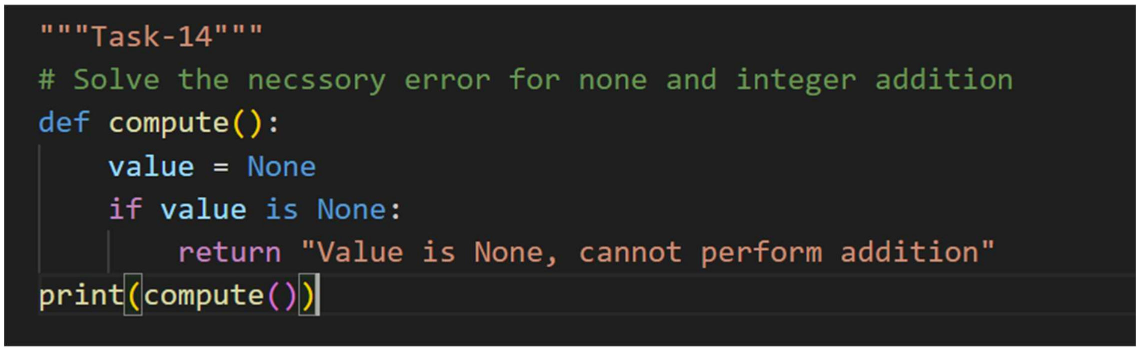
```
# Bug: Adding None and integer
def compute():
    value = None
    return value + 10
print(compute())
```

Requirements:

- Run and identify the error.
- Explain why NoneType cannot be added.
- Fix by assigning a default value.
- Validate using asserts.

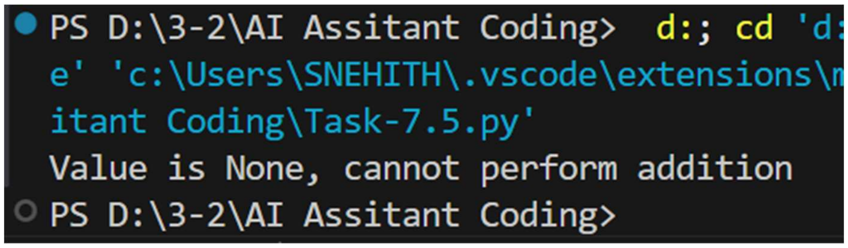
Prompt: # Solve the necssory error for none and integer addition

Code:



```python
"""Task-14"""
# Solve the necssory error for none and integer addition
def compute():
    value = None
    if value is None:
        return "Value is None, cannot perform addition"
print(compute())
```

Output:



```
● PS D:\3-2\AI Assitant Coding>  d:; cd 'd:
  e' 'c:\Users\SNEHITH\.vscode\extensions\m
  itant Coding\Task-7.5.py'
  Value is None, cannot perform addition
○ PS D:\3-2\AI Assitant Coding>
```

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

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

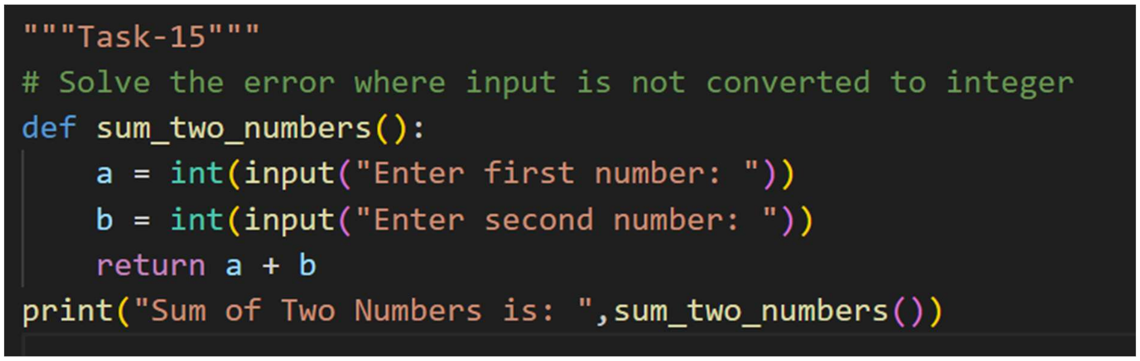
```
# Bug: Input remains string
def sum_two_numbers():
    a = input("Enter first number: ")
    b = input("Enter second number: ")
    return a + b

print(sum_two_numbers())
```

Requirements:

- Explain why input is always string.
- Fix using int() conversion.
- Verify with assert test cases.

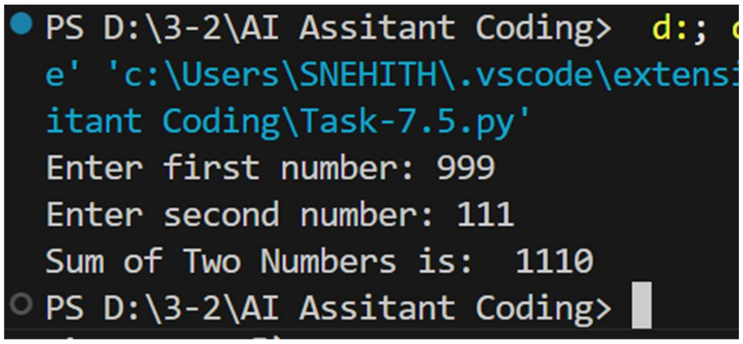
Prompt: # Solve the error where input is not converted to integer

Code:



```
"""Task-15"""
# Solve the error where input is not converted to integer
def sum_two_numbers():
    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))
    return a + b
print("Sum of Two Numbers is: ",sum_two_numbers())
```

Output:



```
PS D:\3-2\AI Assitant Coding>  d:; c
e' 'c:\Users\SNEHITH\.vscode\extensi
itant Coding\Task-7.5.py'
Enter first number: 999
Enter second number: 111
Sum of Two Numbers is:  1110
PS D:\3-2\AI Assitant Coding>
```