AI-ASSISTED CODING

ASSIGNMENT—7.5

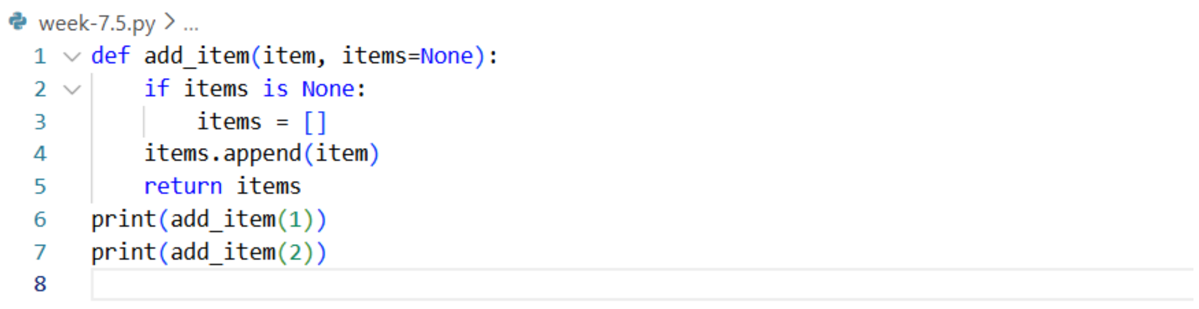
Task 1 (Mutable Default Argument – Function Bug)

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))
```

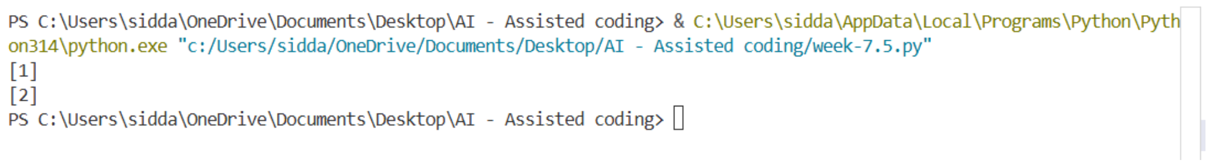
Expected Output: Corrected function avoids shared list bug.

#ai fixed code



```
week-7.5.py > ...
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\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/week-7.5.py"
[1]
[2]
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> 
```

#task 2

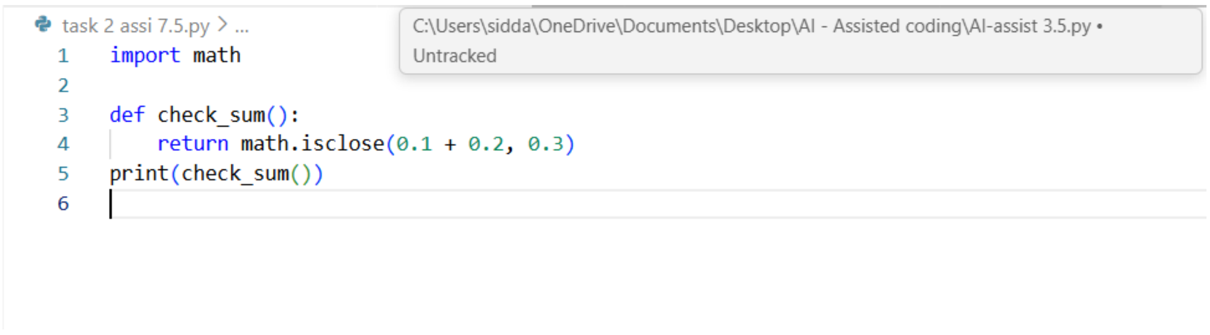
Task 2 (Floating-Point Precision Error)

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

#AI FIXED CODE

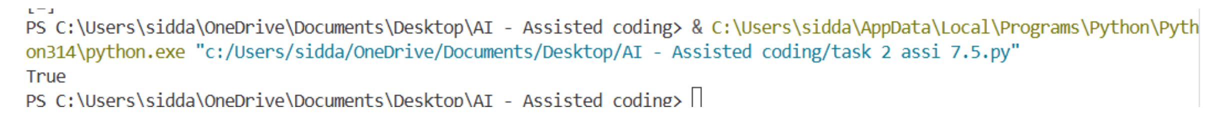


task 2 assi 7.5.py > ...

C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding\AI-assist 3.5.py • Untracked

```python
import math

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

```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 2 assi 7.5.py"
True
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding>
```

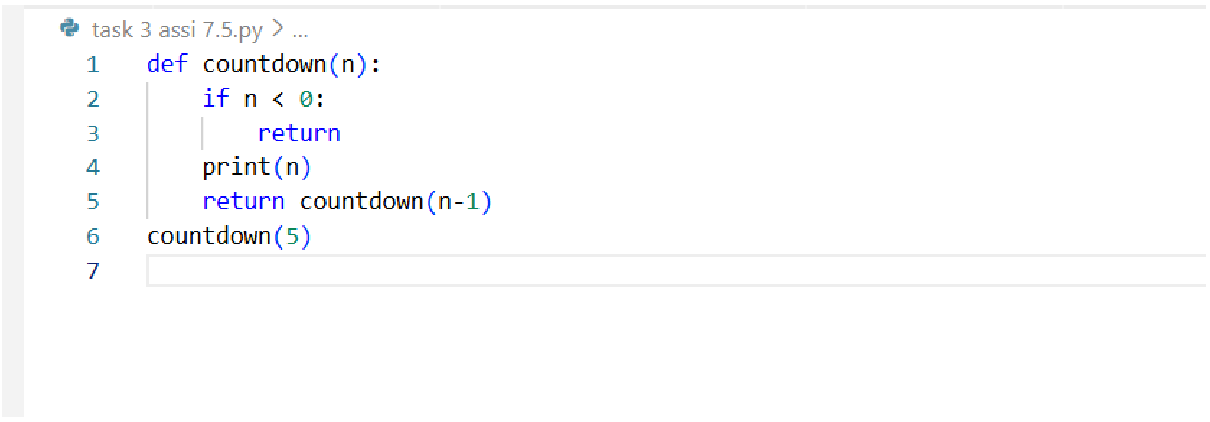
#TASK 3

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.

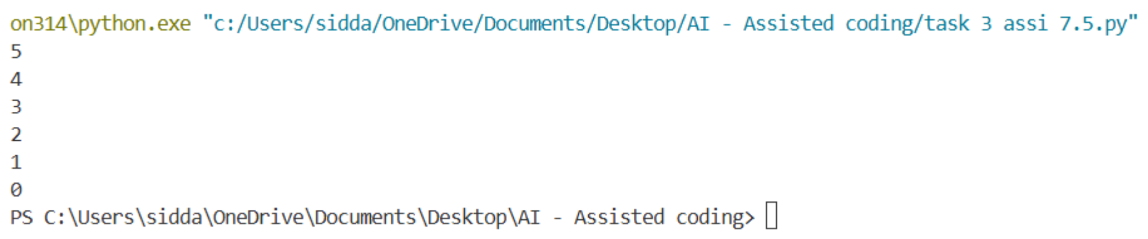
#AI FIXED CODE



task 3 assi 7.5.py > ...

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

```

#OUTPUT



```
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 3 assi 7.5.py"
5
4
3
2
1
0
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding>
```

#TASK4

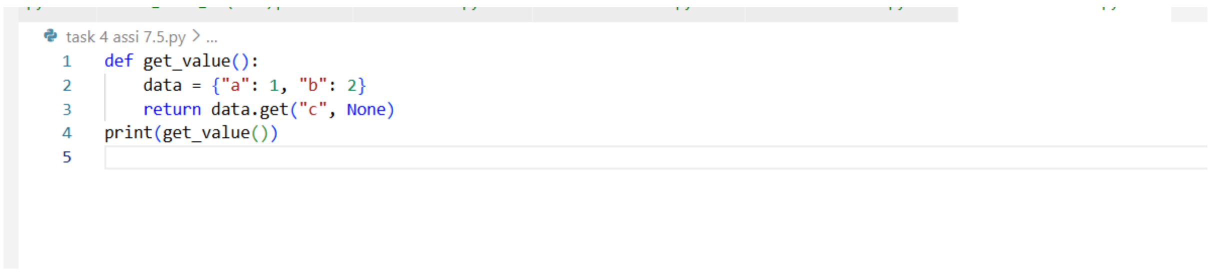
Task 4 (Dictionary Key Error)

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.

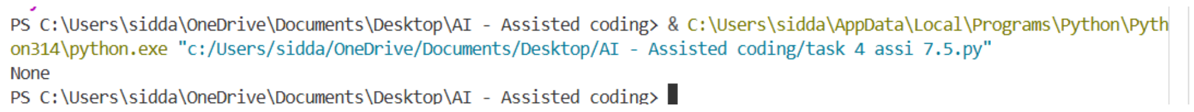
#AI FIXED CODE



task 4 assi 7.5.py > ...

```
def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c", None)
print(get_value())
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
None
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding>
```

#TASK 5

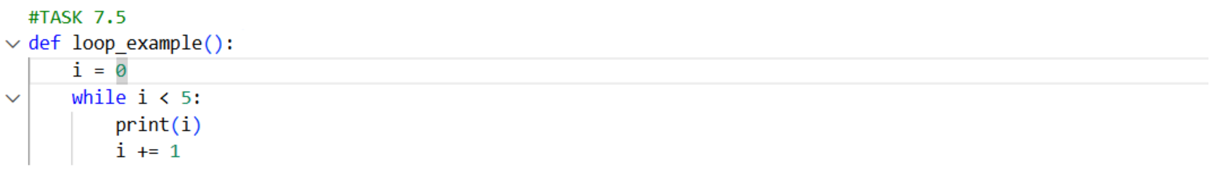
Task 5 (Infinite Loop – Wrong Condition)

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)
```

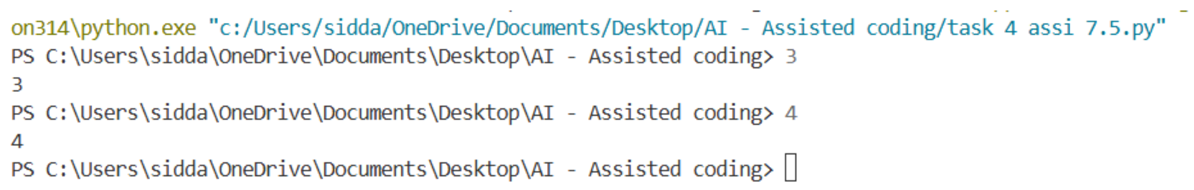
Expected Output: Corrected loop increments i.

#AI-FIXED CODE



```
#TASK 7.5
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
```

#OUT PUT



```
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> 3
3
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> 4
4
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> 
```

#TASK 6

Task 6 (Unpacking Error – Wrong Variables)

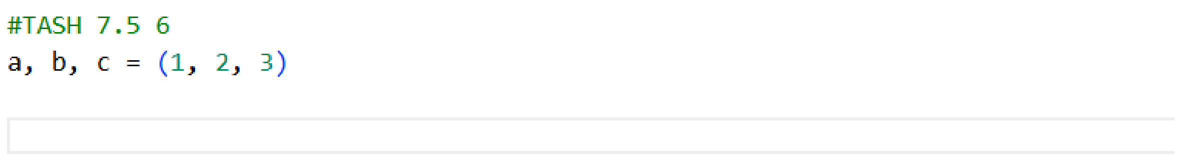
Task: Analyze given code where tuple unpacking fails. Use AI to fix it.

# Bug: Wrong unpacking

a, b = (1, 2, 3)

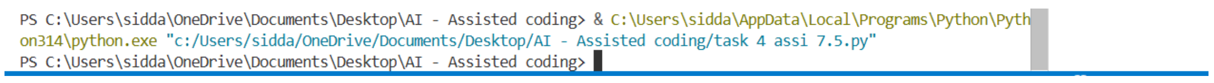
Expected Output: Correct unpacking or using _ for extra values.

#AI-ASSISTED CODE



```
#TASH 7.5 6
a, b, c = (1, 2, 3)
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> 
```

#TASK 7

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():

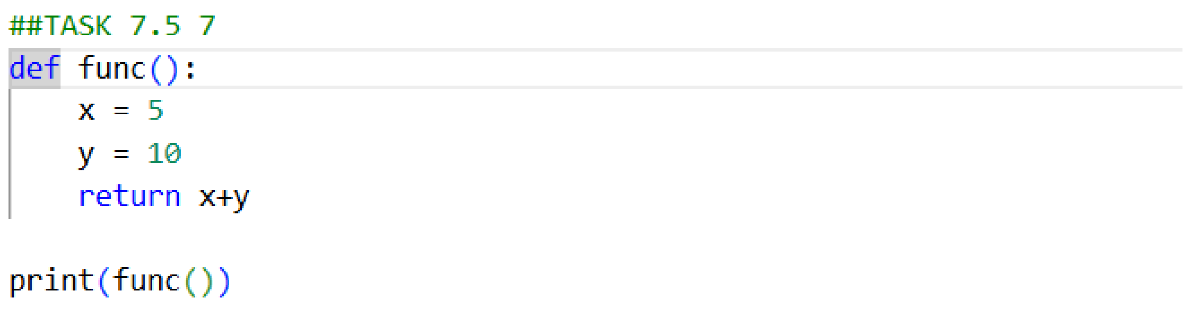
x = 5

y = 10

return x+y

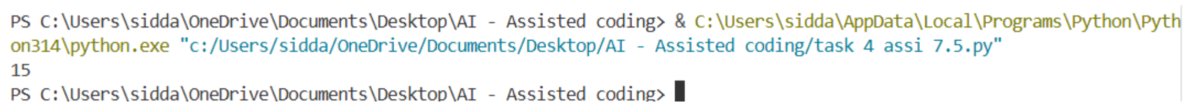
Expected Output : Consistent indentation applied.

#AI-ASSISTED CODE



```
##TASK 7.5 7
def func():
    x = 5
    y = 10
    return x+y

print(func())
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
15
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> 
```

#TASK-8

Task 8 (Import Error – Wrong Module Usage)

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

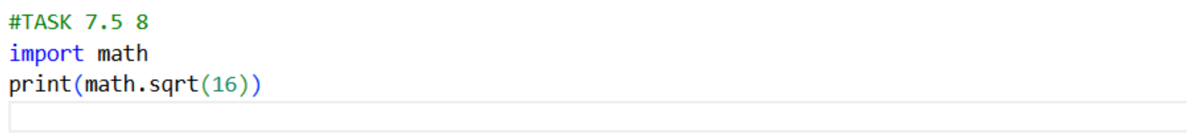
# Bug: Wrong import

import maths

print(maths.sqrt(16))

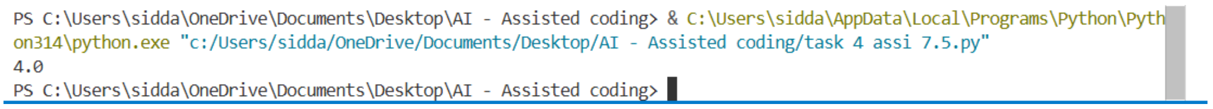
Expected Output: Corrected to import math

#AI-ASSISTED CODE



```
#TASK 7.5 8
import math
print(math.sqrt(16))
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
4.0
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> 
```

#TASK 9

Task 9 (Infinite Loop – Wrong Condition)

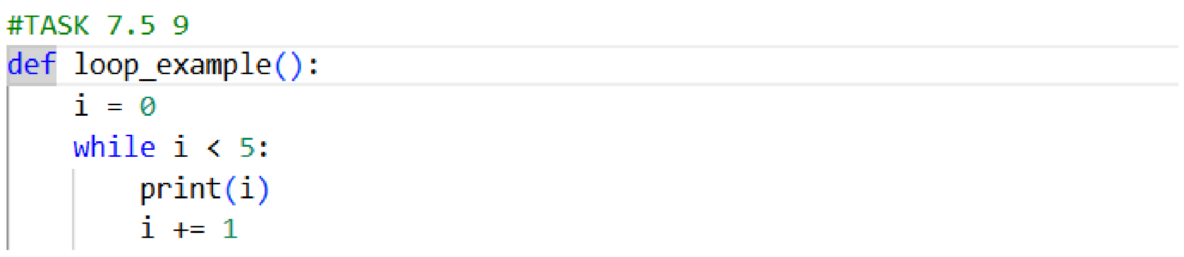
Task: Analyze given code where loop never ends. Use AI to detect and fix it.

# Bug: Infinite loop

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

Expected Output: Corrected loop increments i.

#AI-ASSISTED CODE



```
#TASK 7.5 9
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
```

#TASK-10

Task 9 (Unreachable Code – Return Inside Loop)

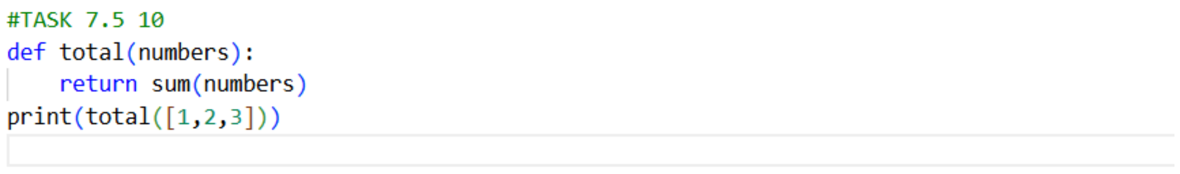
Task: Analyze given code where a return inside a loop prevents full iteration. Use AI to fix it.

# Bug: Early return inside loop

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

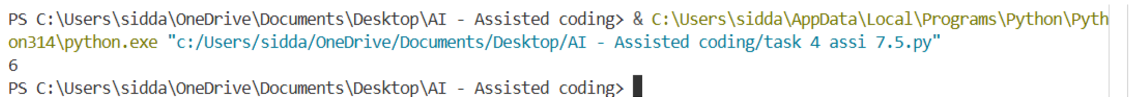
Expected Output: Corrected code accumulates sum and returns after loop.

#AI-ASSISTED CODE



```
#TASK 7.5 10
def total(numbers):
    return sum(numbers)
print(total([1,2,3]))
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
6
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> 
```

#TASK-11

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:

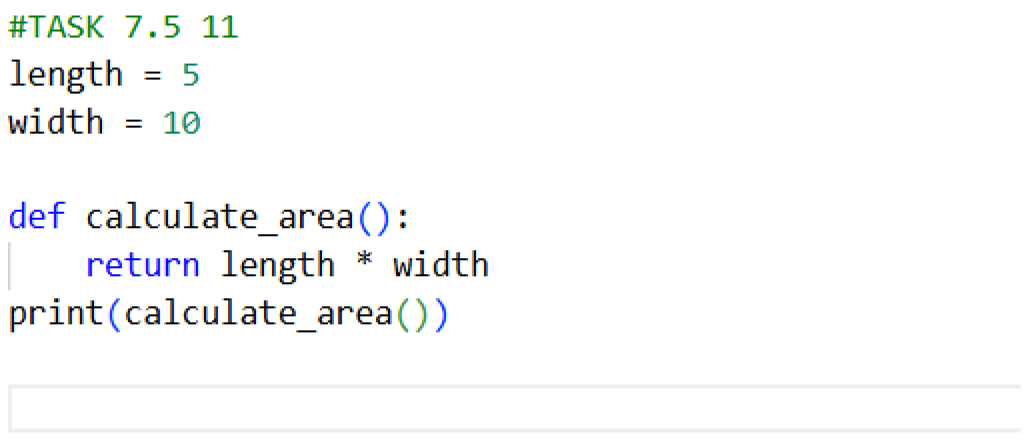
- Run the code to observe the error.
- Ask AI to identify the missing variable definition.
- Fix the bug by defining length and width as parameters.
- Add 3 assert test cases for correctness.

Expected Output :

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

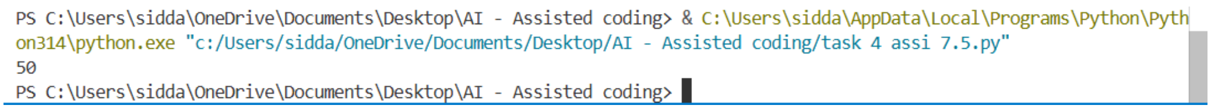
Successful execution of assertions.

#AI-ASSISTED CODE



```
#TASK 7.5 11
length = 5
width = 10

def calculate_area():
    return length * width
print(calculate_area())
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
50
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> 
```

#TASK-12

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:

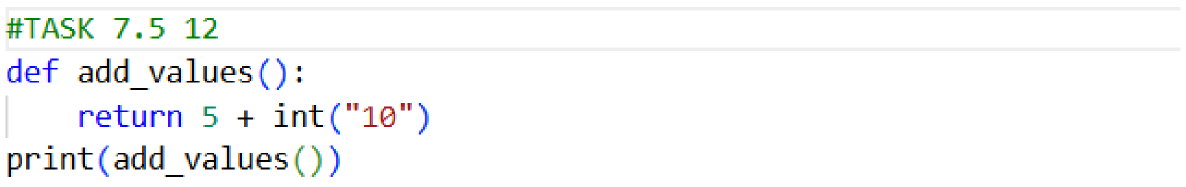
- Run the code to observe the error.
- AI should explain why int + str is invalid.
- Fix the code by type conversion (e.g., int("10") or str(5)).
- Verify with 3 assert cases.

Expected Output #6:

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

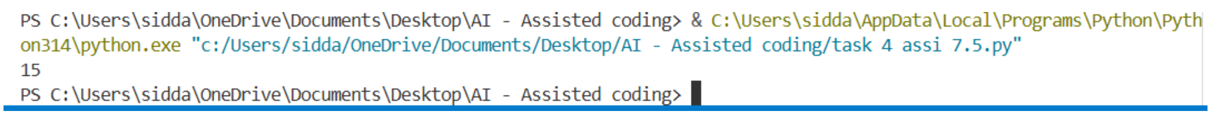
Successful test validation.

#AI-ASSISTED CODE



```
#TASK 7.5 12
def add_values():
    return 5 + int("10")
print(add_values())
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Python314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
15
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding>
```

#TASK-13

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:

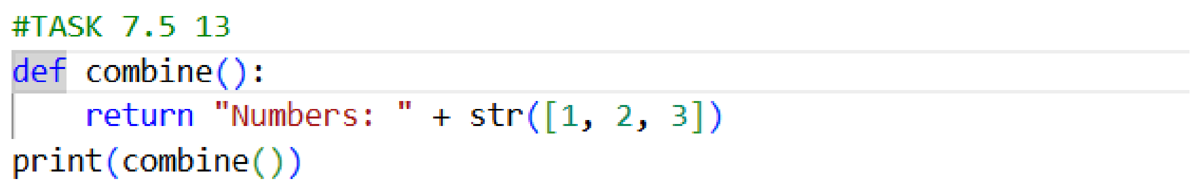
- Run the code to observe the error.
- Explain why str + list is invalid.
- Fix using conversion (str([1,2,3]) or " ".join()).
- Verify with 3 assert cases.

Expected Output:

- Corrected code
- Explanation

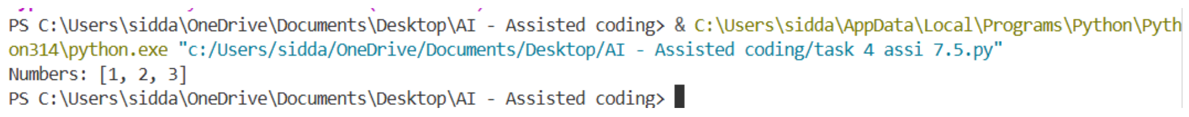
Successful test validation

#AI-ASSISTED CODE



```python
#TASK 7.5 13
def combine():
    return "Numbers: " + str([1, 2, 3])
print(combine())
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
Numbers: [1, 2, 3]
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> ▌
```

#TASK-14

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():

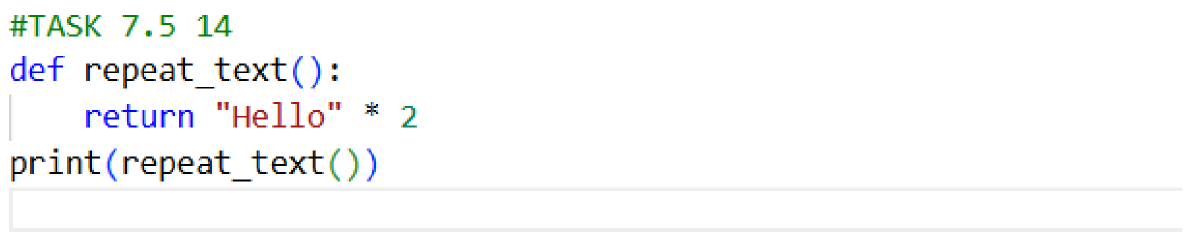
  return "Hello" * 2.5

print(repeat_text())

Requirements:

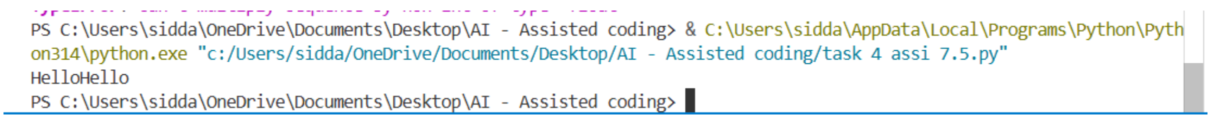
- Observe the error.
- Explain why float multiplication is invalid for strings.
- Fix by converting float to int.
- Add 3 assert test cases.

#AI-ASSISTED CODE



```python
#TASK 7.5 14
def repeat_text():
    return "Hello" * 2
print(repeat_text())
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
HelloHello
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> ▌
```

#TASK-15

Task 14 (Type Error – Adding None to Integer)

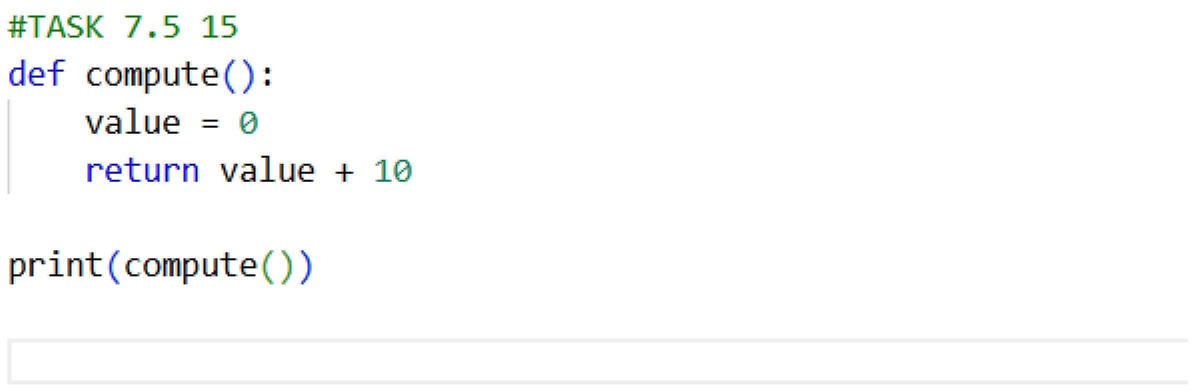
Task: Analyze code where None is added to an integer.

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

print(compute())
```

Requirements:

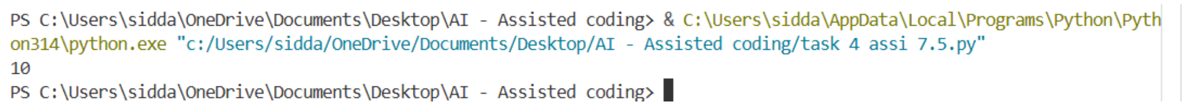
- Run and identify the error.
- Explain why NoneType cannot be added.
- Fix by assigning a default value.
- Validate using asserts.

#AI-ASSISTED CODE



```
#TASK 7.5 15
def compute():
    value = 0
    return value + 10

print(compute())
```

#OUTPUT



```
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
10
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> █
```

#TASK-16

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

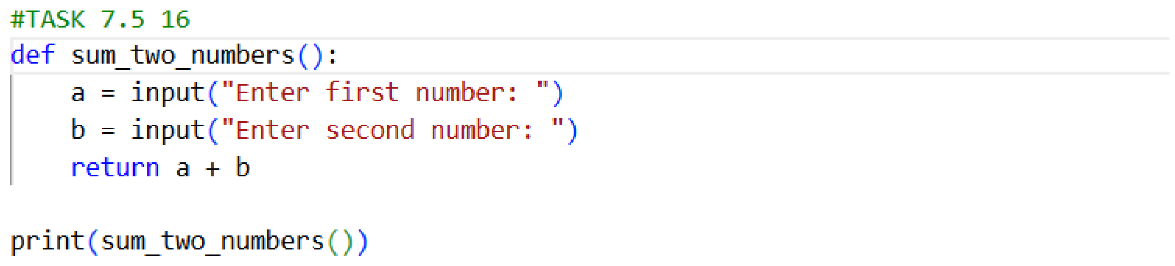
Task: Fix code where user input is not converted properly.

```
# 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:

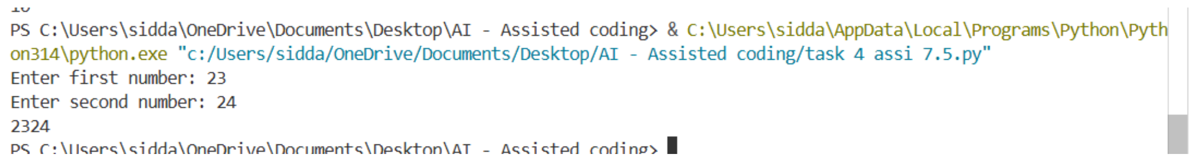
- Explain why input is always string.
- Fix using int() conversion.
- Verify with assert test cases.

#AI-ASSISTED CODE



```python
#TASK 7.5 16
def sum_two_numbers():
    a = input("Enter first number: ")
    b = input("Enter second number: ")
    return a + b

print(sum_two_numbers())
```

#OUTPUT



```
10
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding> & C:\Users\sidda\AppData\Local\Programs\Python\Pyth
on314\python.exe "c:/Users/sidda/OneDrive/Documents/Desktop/AI - Assisted coding/task 4 assi 7.5.py"
Enter first number: 23
Enter second number: 24
2324
PS C:\Users\sidda\OneDrive\Documents\Desktop\AI - Assisted coding>
```