

Assignment-7.4

U.VIGNESH

2303A51964

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.

CODE:

```
asiingment-7.4.py > add_item
1  def add_item(item, items=None):
2      if items is None:
3          items = []
4      items.append(item)
5      return items
6  print(add_item(1))
7  print(add_item(2))
```

OUTPUT:

```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/l
p/AI ASSESTENT CODING/asiingment-7.4.py"
[1]
[2]
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/l
p/AI ASSESTENT CODING/asiingment-7.4.py"
[1]
[2]
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT 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

CODE:

```
❸ asiingment-7.4.py > ...
1  #correct the code so that it returns True
2  def check_sum():
3      return abs((0.1 + 0.2) - 0.3) < 1e-15
4  print(check_sum())
```

OUTPUT:

```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/l
p/AI ASSESTENT CODING/asiingment-7.4.py"
True
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/l
p/AI ASSESTENT CODING/asiingment-7.4.py"
True
```

Task 3 (Recursion Error – Missing Base Case)

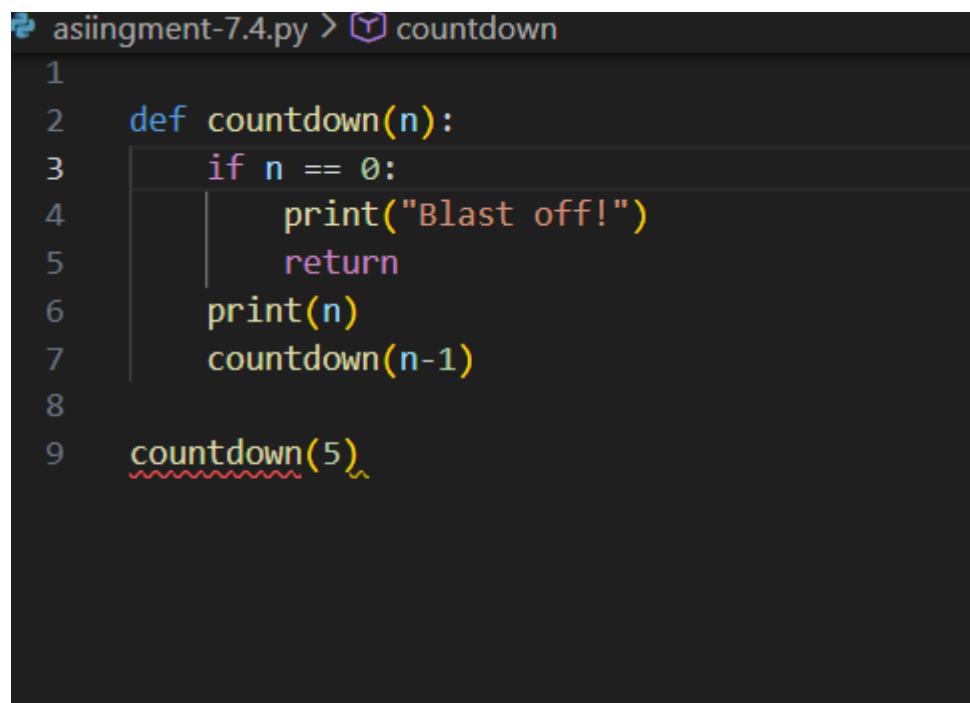
Task: Analyze given code where recursion runs infinitely due to missing base case. Use AI to fix.

```
# Bug: No base case
```

```
def countdown(n):
    print(n)
    return countdown(n-1)
countdown(5)
```

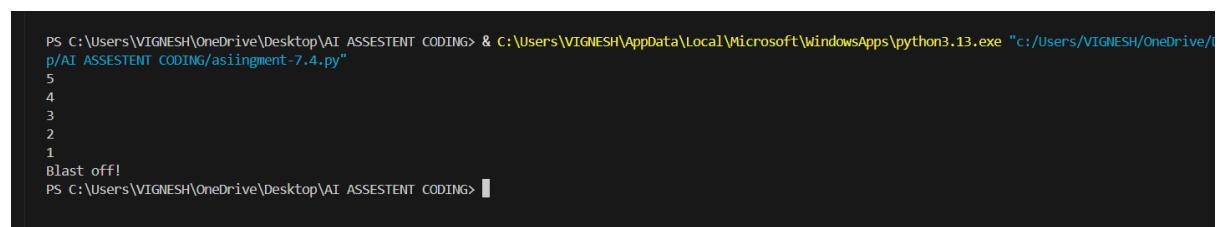
Expected Output : Correct recursion with stopping condition

Code:



```
assignment-7.4.py > countdown
1
2     def countdown(n):
3         if n == 0:
4             print("Blast off!")
5             return
6         print(n)
7         countdown(n-1)
8
9     countdown(5)
```

OUTPUT:



```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESSMENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/Users/VIGNESH/OneDrive/Assignment-7.4.py"
5
4
3
2
1
Blast off!
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESSMENT 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():

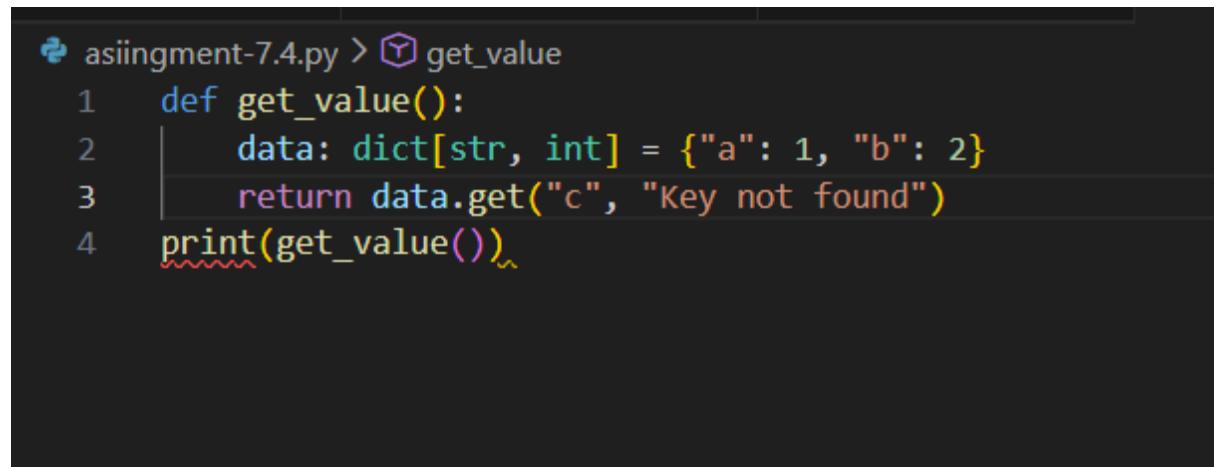
    data = {"a": 1, "b": 2}

    return data["c"]

print(get_value())
```

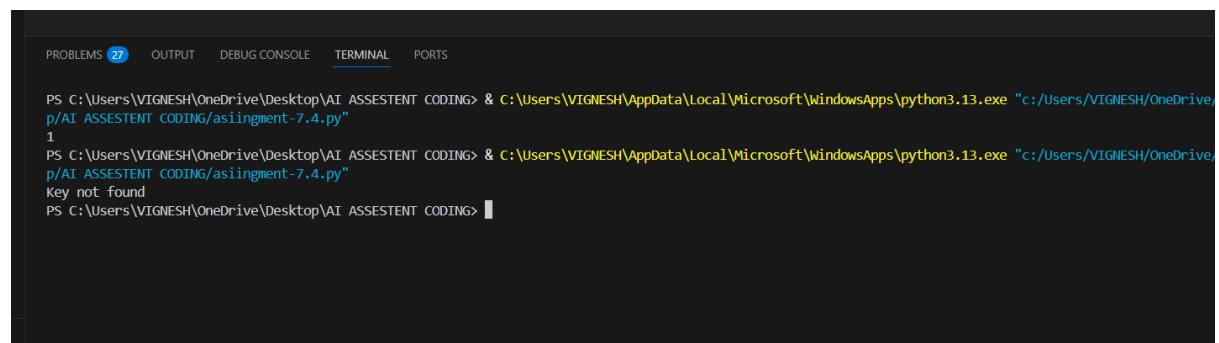
Expected Output: Corrected with .get() or error handling.

Code:



```
assignment-7.4.py > get_value
1 def get_value():
2     data: dict[str, int] = {"a": 1, "b": 2}
3     return data.get("c", "Key not found")
4 print(get_value())
```

Output:



```
PROBLEMS 27 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/Users/VIGNESH/OneDrive/p/AI ASSESTENT CODING/assignment-7.4.py"
1
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/Users/VIGNESH/OneDrive/p/AI ASSESTENT CODING/assignment-7.4.py"
Key not found
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT 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.

Code:

```
1 #correct the code
2 def loop_example():
3     i = 0
4     while i < 5:
5         print(i)
6         i += 1
7 loop_example()
8 ~
9
```

Output:

```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/Users/VIGNESH/OneDrive/Desktop/AI ASSESTENT CODING/assignment-7.4.py"
0
1
2
3
4
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING>
```

Task 6 (Unpacking Error – Wrong Variables)

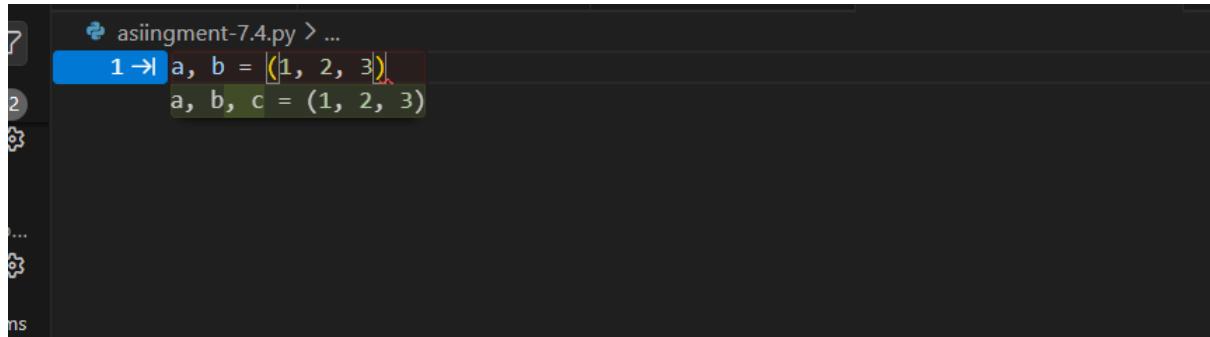
Task: Analyze given code where tuple unpacking fails. Use AI to fix it.

```
# Bug: Wrong unpacking
```

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

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

Code:



```
assignment-7.4.py > ...
1 → a, b = [1, 2, 3]
2   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.

Code:

```
• asiingment-7.4.py > ...
1  def func():
2      x = 5
3      y = 10
4      return x+y
5  result = func()
6  print(result)
7
```

Output:

The screenshot shows a terminal window with the following output:

```
PROBLEMS 29 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING\asiingment-7.4.py"
15
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT 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

Code:

```
asuignment-7.4.py
1 import math
2 print(math.sqrt(16))
```

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

Code:

```
asuignment-7.4.py > ...
1 def total(numbers):
2     total = 0
3     for n in numbers:
4         total += n
5     return total
6 print(total([1,2,3]))
```

Output:

```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\Ap p/AI ASSESTENT CODING/asiingment-7.4.py"
6
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\Ap p/AI ASSESTENT CODING/asiingment-7.4.py"
6
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT 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.

Code:

```
## corrected code taking length and width as parameters and returning the area of a rectangle
def calculate_area(length, width):
    return length * width

print(calculate_area(5, 3))
print(calculate_area(10, 2))
print(calculate_area(7, 4))
```

Output:

PROBLEMS 29 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe p/AI ASSESTENT CODING/asiingment-7.4.py"
15
20
28
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT 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.

Code:

```
assignment7.py > ...
#explain the error in the following code and correct it by type casting
def add_values():
    return 5 + int("10")
print(add_values())
#explanation: The error in the original code is that it tries to add an integer (5) and a string ("10") without type casting.
# This results in a TypeError because Python cannot add an integer and a string directly.
# To fix this, we need to convert the string "10" to an integer using the int() function before performing the addition.
```

```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:\Users\VIGNESH\AppData\Local\Microsoft\WindowsApps\python3.13.exe p/AI ASSESTENT CODING/asiingment-7.4.py"
15
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT 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.

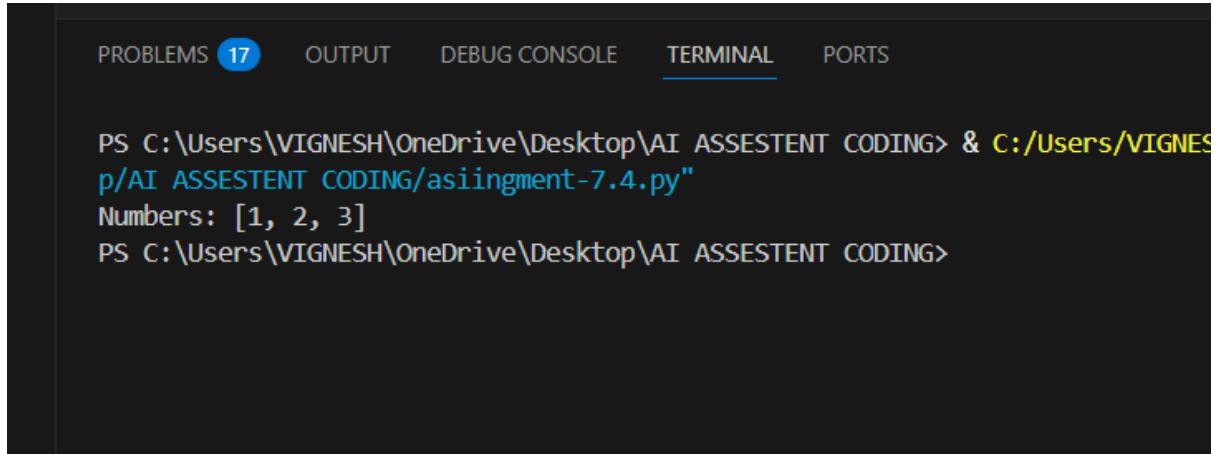
Expected Output:

- Corrected code
- Explanation
- Successful test validation

Code:

```
❶ asiiingment-7.4.py > ...
1  #explain the error in the following code and correct it.
2  def combine():
3      numbers = [1, 2, 3]
4      return "Numbers: " + str(numbers)
5  print(combine())
6
7
8  # The error in the original code is that it tries to concatenate a string with a list, which is not allowed in Python.
9  # To fix this, we need to convert the list of numbers into a string before concatenating it.
10 # This can be done using the `str()` function to type cast the list into a string. The corrected code is as follows:
```

Output:



The screenshot shows a terminal window with the following output:

```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING & C:/Users/VIGNESH/OneDrive/Desktop/AI ASSESTENT CODING/assignment-7.4.py"
Numbers: [1, 2, 3]
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT 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():
```

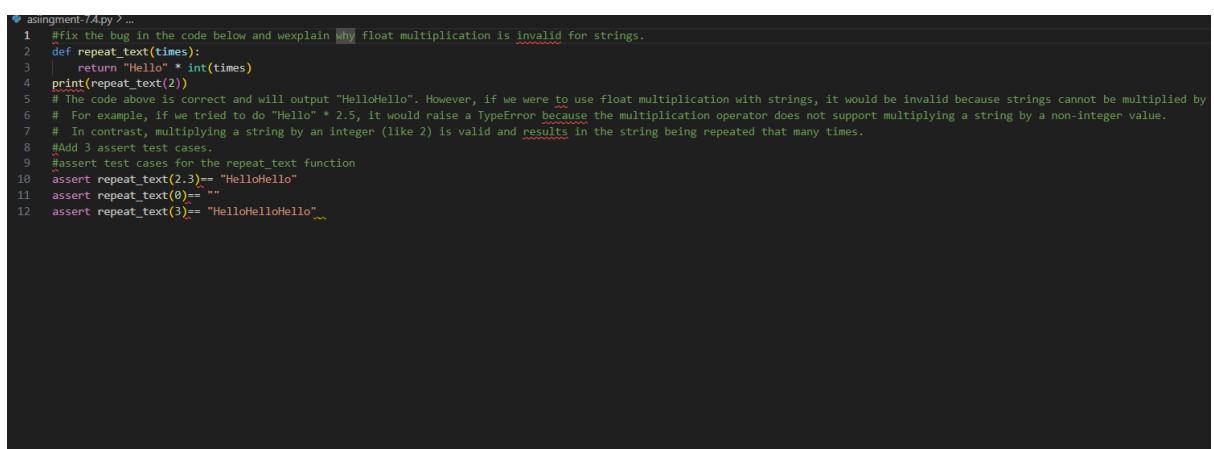
```
    return "Hello" * 2.5
```

```
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.

Code:



```
assignment-7A.py > ...
1 #fix the bug in the code below and wexplain why float multiplication is invalid for strings.
2 def repeat_text(times):
3     return "Hello" * int(times)
4 print(repeat_text(2))
5 # The code above is correct and will output "HelloHello". However, if we were to use float multiplication with strings, it would be invalid because strings cannot be multiplied by floating point numbers.
6 # For example, if we tried to do "Hello" * 2.5, it would raise a TypeError because the multiplication operator does not support multiplying a string by a non-integer value.
7 # In contrast, multiplying a string by an integer (like 2) is valid and results in the string being repeated that many times.
8 #Add 3 assert test cases.
9 #assert test cases for the repeat_text function
10 assert repeat_text(2.3) == "HelloHello"
11 assert repeat_text(0) == ""
12 assert repeat_text(3) == "HelloHelloHello"...
```

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

Code:

```
1  #correct the code explain why NoneType cannot be added.
2  def compute(value=None):
3      if value is None:
4          value = 0
5      return value + 10
6  print(compute())  # Output: 10
7  print(compute(5)) # Output: 15
8  # Assert test cases
9  assert compute() == 10
10 assert compute(5) == 15
11 assert compute(0) == 10
12 # Explanation:
13 # In the original code, if the default value of 'value' is None, it cannot
14 # be added to 10 because NoneType cannot be added to an integer. This would
15 # result in a TypeError. By checking if 'value' is None and assigning it a default value of 0,
16 # we ensure that the function can perform the addition without errors.
```

Output:

```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING & C:/Users/VIGNESH/AppData/Local/Microsoft/Windows/Desktop/AI ASSESTENT CODING/asiingment-7.4.py"
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING & C:/Users/VIGNESH/AppData/Local/Microsoft/Windows/Desktop/AI ASSESTENT CODING/asiingment-7.4.py"
10
15
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT 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.

Code:

```
▶ asignment-7.4.py > sum_two_numbers
1 #correct the code and explain why input is always string.
2 def sum_two_numbers():
3     a = input("Enter first number: ")
4     b = input("Enter second number: ")
5     5
6     return int(a) + int(b)
7     print(sum_two_numbers())
8     # The input function in Python always returns a string, regardless of what the user enters.
9     # This is because the input function is designed to read a line of text from the user, and it treats all input as text.
10    # Therefore, when we want to perform arithmetic operations on the input, we need to convert the string to an integer (or float) using the int() or f
11    # In this code, we convert the inputs 'a' and 'b' to integers before adding them together, which allows us to get the correct sum of the two numbers
```

Output:

```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING> & C:/Users/VIGNESH/AppData/Local/Microsoft/Desktop/AI ASSESTENT CODING/asiingment-7.4.py"
Enter first number: 5
Enter second number: 4
9
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING>
```