## Name: V. VIGNESH

## H.no:2303A51707

## Batch:24

# 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.**

**INPUT:**

```
#fix the error in the following code snippet and print the output.

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

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
[1]
[1, 2]
Task 1 completed
```

**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**

**INPUT:**

```
#fix the error in the following code snippet and print the output.

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

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
False
Task 2 is completed
```

**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.**

**INPUT:**

```python
#fix the error in the following code snippet and print the output.

def countdown(n):
    print(n)
    if n == 0:
        return
    return countdown(n-1)
countdown(5)
print("Task 3 Completed")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
5
4
3
2
1
0
Task 3 Completed
```

**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.**

**INPUT:**

```python
#fix the error in the following code snippet and print the corrected output.

def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c", "Key not found")
print(get_value())
print("Task 4 completed successfully.")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
Key not found
Task 4 completed successfully.
```

**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.**

**INPUT:**

```python
#fix the error in the following code snippet and print the corrected output.

def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
loop_example()
print("Task 5 completed successfully.")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
0
1
2
3
4
Task 5 completed successfully.
```

**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.**

**INPUT:**

```
ssn_7.4.py > ...
  #fix the error in the following code snippet and print the corrected output.

  #a, b = (1, 2, 3)
  # The error is that there are more values on the right side than variables on the left side.
  # To fix this, we can either reduce the number of values on the right side or increase
  # the number of variables on the left side.
  a, b, c = (1, 2, 3)
  print(a, b, c)
  print("Task 6 is completed.")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
1 2 3
Task 6 is completed.
```

**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.**

**INPUT:**

```
assn_7.4.py > ...
1  #fix the error in the following code snippet and print the Consistent indentation applied. output.
2
3  def func():
4      x = 5
5      y = 10
6      return x+y
7
8  print(func())
9  print("Task 7 is completed with consistent indentation applied.")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
15
Task 7 is completed with consistent indentation applied.
```

**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**

**INPUT:**

```
n_7.4.py > ...                C:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\cummulative_sums.py
#fix the error in the following code snippet and print the output.

import math
print(math.sqrt(16))
print("Task 8 completed successfully.")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
4.0
Task 8 completed successfully.
```

**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.**

**INPUT:**

```
#fix the error in the following code snippet and print the output.

def total(numbers):
    total_sum = 0
    for n in numbers:
        total_sum += n
    return total_sum

print(total([1,2,3]))
print("Task 9 completed successfully.")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
6
Task 9 completed successfully.
```

**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.**

**INPUT:**

```
#fix the error in the following code snippet and print the output.

def calculate_area(length, width):
    area = length * width
    return area
length = 5
width = 3
area = calculate_area(length, width)
print("The area of the rectangle is:", area)# The code is correct and should run without any errors. When you run the code, it will
print("Task 10 is completed successfully.")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
The area of the rectangle is: 15
Task 10 is completed successfully.
```

**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.**

**INPUT:**

```
#fix the error in the following code snippet and print the output.

def add_values():
    return str(5) + "10"
print(add_values())
print("Task 11 completed successfully!")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
510
Task 11 completed successfully!
```

**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())**

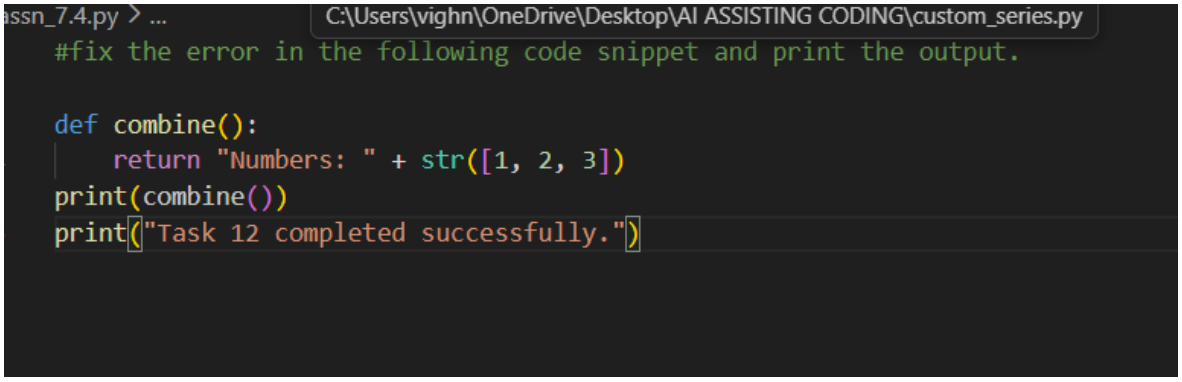
**Requirements:**

- **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.**

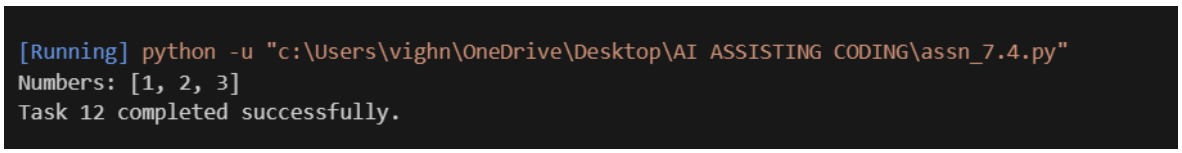
**Expected Output:**

- **Corrected code**
- **Explanation**
- **Successful test validation**

**INPUT:**




**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.**

**INPUT:**

```python
#fix the error in the following code snippet and print the output.

def repeat_text():
    return "Hello" * 2

print(repeat_text())
print("Task 13 completed")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
HelloHello
Task 13 completed
```

**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.**

**INPUT:**

```python
n_7.4.py > ...
#fix the error in the following code snippet and print the output.

def compute():
    value = 0
    return value + 10

print(compute())
print("Task 14 completed")
```

**OUTPUT:**

```
[Running] python -u "c:\Users\vighn\OneDrive\Desktop\AI ASSISTING CODING\assn_7.4.py"
10
Task 14 completed
```

**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.**

**INPUT:**

```python
#fix the error in the following code snippet and print the output.

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

print(sum_two_numbers())
```

**OUTPUT:**

```
vighn@Vighnesh MINGW64 ~/OneDr
$ python assn_7.4.py
Enter first number: 12
Enter second number: 23
35
```