

ASSIGNMENT-7.5

Name: V. Sravani

Ht.No: 2303A51923

Batch:23

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.

```
9  #don't use mutable default arguments
10 def add_item(item, items=None):
11     if items is None:
12         items = []
13     items.append(item)
14     return items
15
16 print(add_item(1,[10,20]))
17 print(add_item(2,[15,30]))
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5> & C:/Python364/python.exe "c:/Users/SRAVANI/Documents/AI Assist/Assignment-7.5.py"
[10, 20]
[15, 30]
```

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

```
11  def check_sum():  
12      return abs(0.1 + 0.2) - 0.3) < 1e-9  
13  
14  print(check_sum())  
15  
16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.  
:/Users/SRAVANI/Documents/AI Assist/Assignment-7.5/02  
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.

```
6
7     def countdown(n):
8         if n <= 0:
9             print(0)
10            return
11         print(n)
12         countdown(n - 1)
13
14     countdown(5)
15 
```

Output (Ctrl+Shift+U)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5> & C:/Users/SRAVANI/Documents/AI Assist/Assignment-7.5/03.py
5
4
3
2
1
0
```

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.

```
7     #Fix the error in the code
8     def get_value():
9         data = {"a": 1, "b": 2}
10        try:
11            return data["c"]
12        except KeyError:
13            return ("key not found")
14
15     print(get_value())
16 
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5> & C:/Users/SRAVANI/Documents/AI Assist/Assignment-7.5/03.py
key not found
```

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.

The screenshot shows a code editor interface with a dark theme. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is currently selected, indicated by an underline. Below the tabs, the terminal window displays the following text:

```
PS C:\Users\SRAVANI\Documents\AI Assist\A
:/Users/SRAVANI/Documents/AI Assist/Assig
0
1
2
3
4
```

The code editor itself has the following content:

```
7     def loop_example():
8         i = 0
9         while i < 5:
10            print(i)
11            i += 1
12
13 loop_example()
14
15
```

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.

```
13     a, b, c = (1, 2, 3)
14     print(a,b,c)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assig
:/Users/SRAVANI/Documents/AI Assist/Assig
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.

```
7     #Fix the Indentation
8     def func():
9         x = 5
10        y = 10
11        return x + y
12    result=func()
13    print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assig
:/Users/SRAVANI/Documents/AI Assist/Assig
15
```

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

The screenshot shows a terminal window with the following content:

```
8: # import correctly  
9: import math  
10: print(math.sqrt(16))  
  
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7  
4.0
```

The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORT.

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.

The screenshot shows a terminal window with the following content:

```
9: def total(numbers):  
10:     total = 0  
11:     for n in numbers:  
12:         total += n  
13:     return total  
14:  
15: print(total([1, 2, 3]))  
16  
  
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7  
4.0
```

The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORT.

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.

```
6  #function to calculate the area of a rectangle  
7  def calculate_area(length, width):  
8      #multiply length and width to get the area  
9      return length * width  
10     #call the function with example values  
11     length = 5  
12     width = 3  
13     area = calculate_area(length, width)  
14     print(f"The area of the rectangle is: {area}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5> & C  
:/Users/SRAVANI/Documents/AI Assist/Assignment-7.5/10.py"  
The area of the rectangle is: 15
```

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.

The screenshot shows a terminal window with the following content:

```
10  def add_values():
11      return 10 + 10
12  print(add_values())
13

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\SRAVANI\Documents\AI Assist
:/Users/SRAVANI/Documents/AI Assist/Ass
20
PS C:\Users\SRAVANI\Documents\AI Assist
```

The code in the editor has a syntax error at line 11, where the addition operator '+' is used between an integer and a string. The terminal shows the command PS C:\Users\SRAVANI\Documents\AI Assist followed by the path to the file. The output shows the command being run and the resulting error message '20' (likely a placeholder for an error code), indicating that the code did not execute successfully due to the type error.

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

The screenshot shows a terminal window with the following content:

```
9     def combine():  
10    numbers = [1, 2, 3]  
11    return f"Numbers: {numbers}"  
12    print(combine())
```

Below the code, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORT. The terminal output shows:

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7  
:/Users/SRAVANI/Documents/AI Assist/Assignment-7  
Numbers: [1, 2, 3]
```

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

```
4  # str * float is invalid because Python cannot multiply a string by a float
5  # The * operator for strings only works with integers to repeat the string
6  # You must convert the float to an integer first using int()
7  def repeat_text():
8      # Fix: Convert float to int
9      return "Hello" * int(2.5)
10 print(repeat_text())
11 # Verify with 3 assert cases
12 assert repeat_text() == "HelloHello", "Test 1 failed"
13 assert isinstance(repeat_text(), str), "Test 2 failed"
14 assert len(repeat_text()) == 10, "Test 3 failed"
15 print("All assertions passed!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5> & C:/Users/SRAVANI/AppData/Local/Temp/Temporary Items/Python/Assignment-7.5/13.py
HelloHello
All assertions passed!
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5>
```

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.

```

4  def compute():
5      value = 0 # Assign a default value
6      return value + 10
7  result = compute()
8  print(result)
9  # Validate using asserts
10 assert result == 10, "Test 1 failed"
11 assert isinstance(result, int), "Test 2 failed"
12 assert result > 0, "Test 3 failed"
13 print("All assertions passed!")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5> & C:
:/Users/SRAVANI/Documents/AI Assist/Assignment-7.5/14.py"
10
All assertions passed!
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5>

```

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.

```
10     def sum_two_numbers():  
11         a = int(input("Enter first number: "))  
12         b = int(input("Enter second number: "))  
13         return a + b  
14     print(sum_two_numbers())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5> & C:\Python37\python.exe "C:/Users/SRAVANI/Documents/AI Assist/Assignment-7.5/15.py"  
Enter first number: 5  
Enter second number: 20  
25  
PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5>
```