

AI ASSISTED CODING

ASSIGNMENT-7.5

Name: Hari Priya

H.T.No: 2303A51104

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=[]):
```

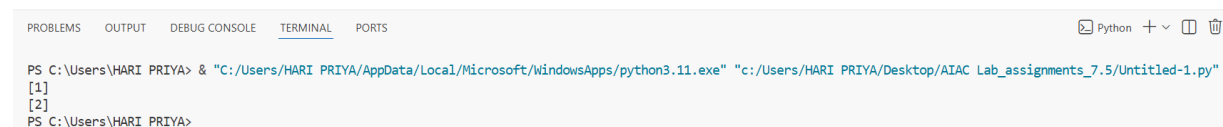
```
items.append(item) return
```

```
items print(add_item(1))
```

```
print(add_item(2))
```

Expected Output: Corrected function avoids shared list bug.

```
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))
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v [ ] [ ]
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-1.py"
[1]
[2]
PS C:\Users\HARI PRIYA>
```

Task 2 (Floating-Point Precision Error)

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

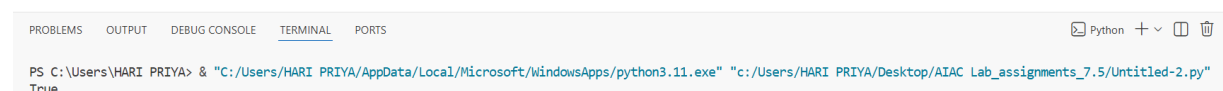
Bug: Floating point precision issue

```
def check_sum(): return (0.1 + 0.2)
```

```
== 0.3 print(check_sum())
```

Expected Output: Corrected function

```
1 import math
2 def check_sum():
3     return math.isclose(0.1 + 0.2, 0.3)
4 print(check_sum())
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v [ ] [ ]
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-2.py"
True
```

Task 3 (Recursion Error – Missing Base Case)

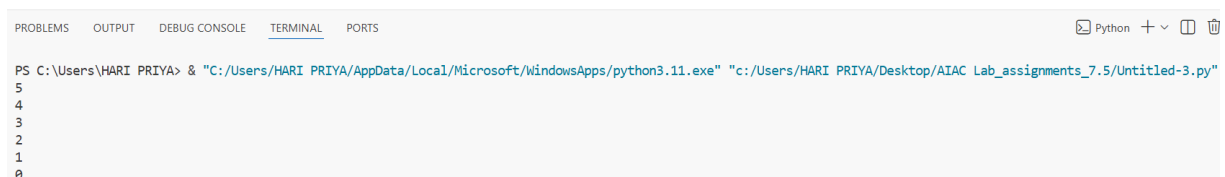
Task: Analyze given code where recursion runs infinitely due to missing base case. Use AI to fix.

Bug: No base case def

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

Expected Output : Correct recursion with stopping condition.

```
1 def countdown(n):  
2     if n < 0:  
3         return  
4     print(n)  
5     return countdown(n-1)  
6     countdown(5)
```



The screenshot shows a Python IDE with a terminal window. The terminal displays the command to run the script and the output of the corrected countdown function, which prints the numbers 5, 4, 3, 2, 1, and 0 in descending order.

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-3.py"  
5  
4  
3  
2  
1  
0
```

Task 4 (Dictionary Key Error)

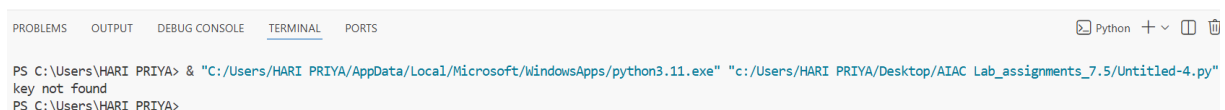
Task: Analyze given code where a missing dictionary key causes error. Use AI to fix it.

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.

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



The screenshot shows a Python IDE with a terminal window. The terminal displays the command to run the script and the output of the corrected get_value function, which prints "key not found" because the key "c" is not present in the dictionary.

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-4.py"  
key not found  
PS C:\Users\HARI PRIYA>
```

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.

```
1 def loop_example():
2     i = 0
3     while i < 5:
4         print(i)
5         i+=1
6 loop_example()
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [X]
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-5.py"
0
1
2
3
4
```

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.

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

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [X]
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-6.py"
PS C:\Users\HARI PRIYA>
```

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.

```
1 def func():
2     x = 5
3     y = 10
4     return x+y
5 print(func())
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [X]
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-6.py"
15
PS C:\Users\HARI PRIYA>
```

Task 8 (Import Error – Wrong Module Usage)

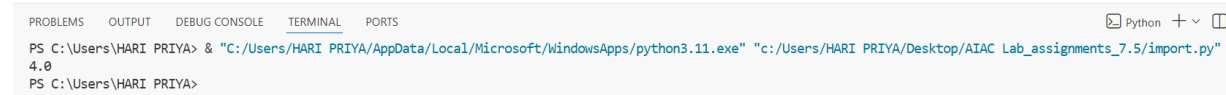
Task: Analyze given code with incorrect import. Use AI to fix.

Bug: Wrong import import

```
maths print(maths.sqrt(16))
```

Expected Output: Corrected to import math

```
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

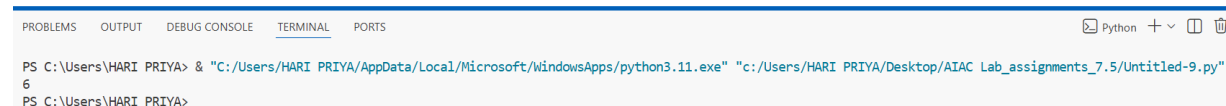
```
def total(numbers): for n in
```

```
numbers: return n
```

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

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

```
1 def total(numbers):
2     s=0
3     for n in numbers:
4         s+=n
5     return s
6 print(total([1,2,3]))
```



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.

```

1  # Function to calculate the area of a rectangle
2  def calculate_area(length,width):
3      # Multiply length and width to get area
4      return length * width
5  # Call the function with length=5 and width=10, then print the result
6  print(calculate_area(5,10))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [] [X]

PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-10.py"

50

PS C:\Users\HARI PRIYA>

```

1  from 2303A51104 Lab Assignment 7 5 import calculate_area
2  def test_calculate_area():
3      assert calculate_area(5, 10) == 50
4      assert calculate_area(3, 4) == 12
5      assert calculate_area(0, 5) == 0
6  test_calculate_area()

```

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.

```
1 def add_values():
2     # Indent the return statement to be inside the function body
3     # Convert the string "10" to an integer before adding
4     return 5 + int("10")
5 # Call the function and print the result
6 print(add_values())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + -

PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-11.py"

15

PS C:\Users\HARI PRIYA>

```
1 from 2303A51104 Lab Assignment 7 5 import add values
2 def test_add_values():
3     assert add_values() == 15
4     assert add_values() == 5 + int("10")
5     assert isinstance(add_values(),int)
6 test_add_values()
7 print("All tests passed!")
```

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

```

1  # str + list is invalid because Python cannot concatenate a string with a list directly
2  # Strings and lists are different types, and the + operator doesn't know how to combine them
3  # You must convert the list to a string first using str() or join()
4  def combine():
5      # Fix: Convert list to string using str()
6      return "Numbers: " + str([1, 2, 3])
7  print(combine())
8  # verify with 3 assert cases
9  assert combine() == "Numbers: [1, 2, 3]", "Test 1 failed"
10 assert isinstance(combine(), str), "Test 2 failed"
11 assert "Numbers:" in combine(), "Test 3 failed"
12 print("All assertions passed!")

```

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

Python
+
-
📄
🗑️

```

PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-12.py"
Numbers: [1, 2, 3]
All assertions passed!
PS C:\Users\HARI PRIYA>

```

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

```

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

```

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

Python
+
-
📄
🗑️

```

PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-13.py"
HelloHello
All assertions passed!

```

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.

```

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

```

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

Python
+
-
🔍
🗑️

```

PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-14.py"
10
All assertions passed!

```

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.

```

1  def sum_two_numbers():
2      a = int(input("Enter first number: ")) # Convert input to int
3      b = int(input("Enter second number: ")) # Convert input to int
4      return a + b
5  result = sum_two_numbers()
6  print(result)
7  # Verify with assert test cases
8  assert isinstance(result, int), "Result should be an integer"
9  assert result == (int(input("Enter first number: ")) + int(input("Enter second number: "))), "Sum does not match expected value"
10 print("All assertions passed!")

```

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

Python
+
-
🔍
🗑️

```

PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/AIAC Lab_assignments_7.5/Untitled-15.py"
Enter first number: 10
Enter second number: 20
30

```