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

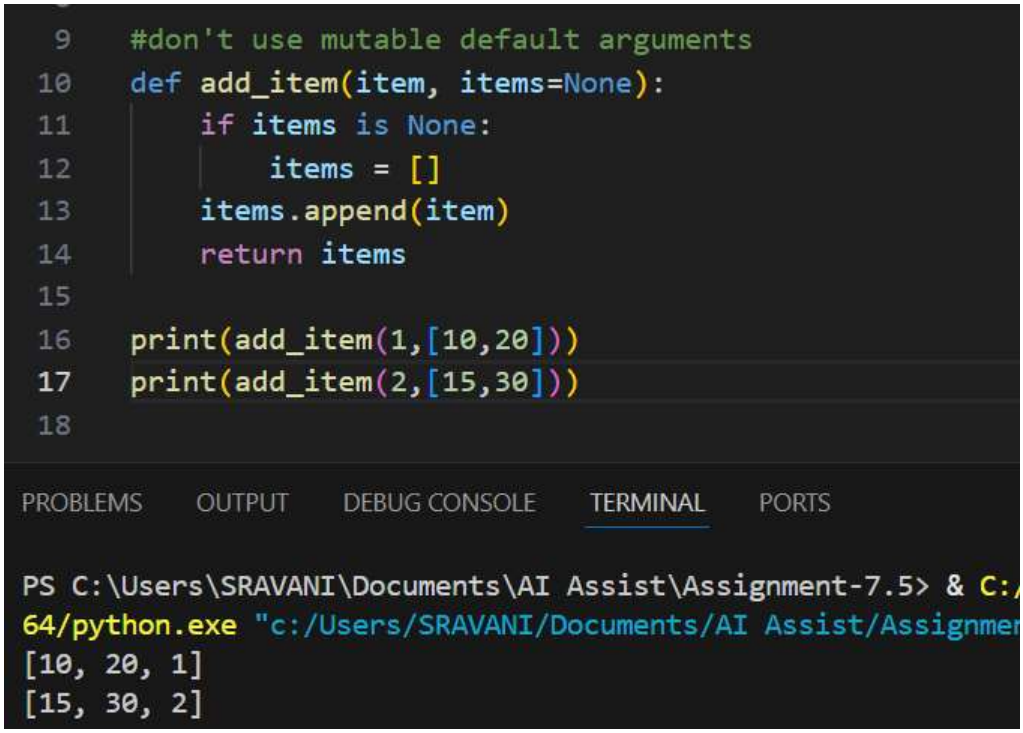
add_item(item, items=[]):

items.append(item) return

items print(add_item(1))

print(add_item(2))

Expected Output: Corrected function avoids shared list bug.



Task 2 (Floating-Point Precision Error)

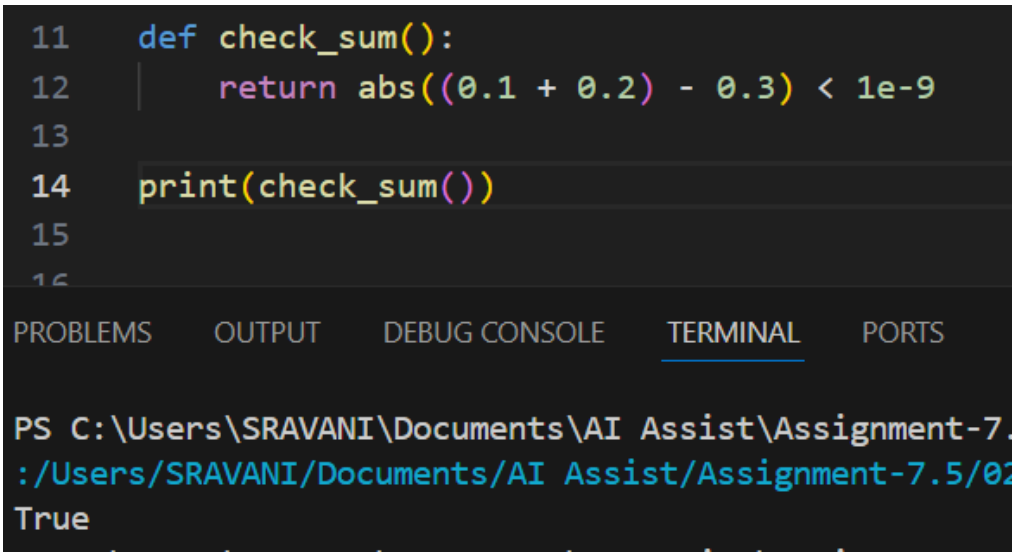
Task: Analyze given code where floating-point comparison fails. Use AI to correct with tolerance.

Bug: Floating point precision issue def

check_sum(): return (0.1 + 0.2)

== 0.3 print(check_sum())

Expected Output: Corrected function



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.

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

countdown(5)
```

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.

```python
#Fix the error in the code
def get_value():
    data = {"a": 1, "b": 2}
    try:
        return data["c"]
    except KeyError:
        return ("key not found")

print(get_value())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

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

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

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

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  TERM

PS C:\Users\SRAVANI\Documents\AI Assis
:/Users/SRAVANI/Documents/AI Assist/As
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\Ass
:/Users/SRAVANI/Documents/AI Assist/Assignm
15
```

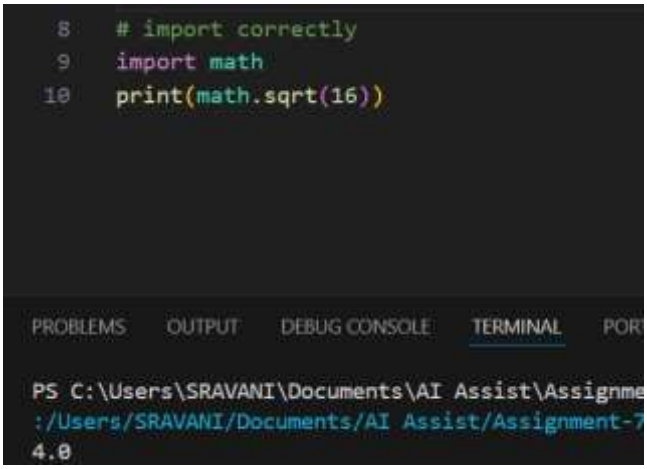
Task 8 (Import Error – Wrong Module Usage)

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



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.

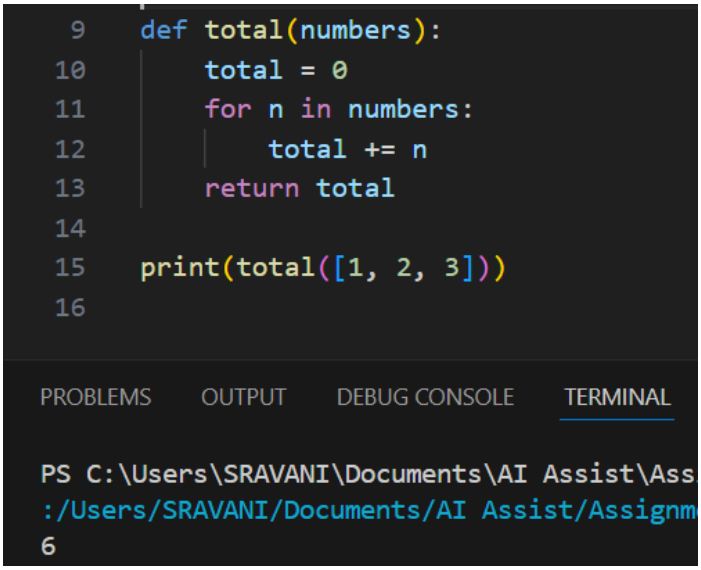
Bug: Early return inside loop def

total(numbers): for n in numbers:

return n

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

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



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)).

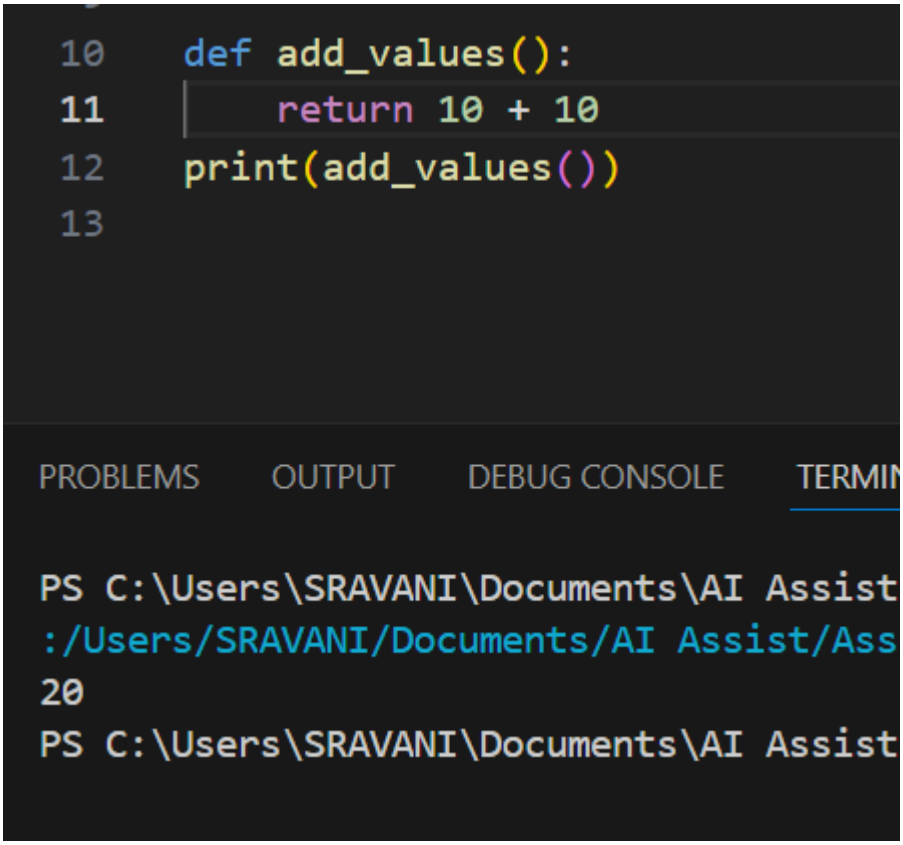
• Verify with 3 assert cases.

Expected Output #6:

• Corrected code with type handling.

• AI explanation of the fix.

Successful test validation.



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()).

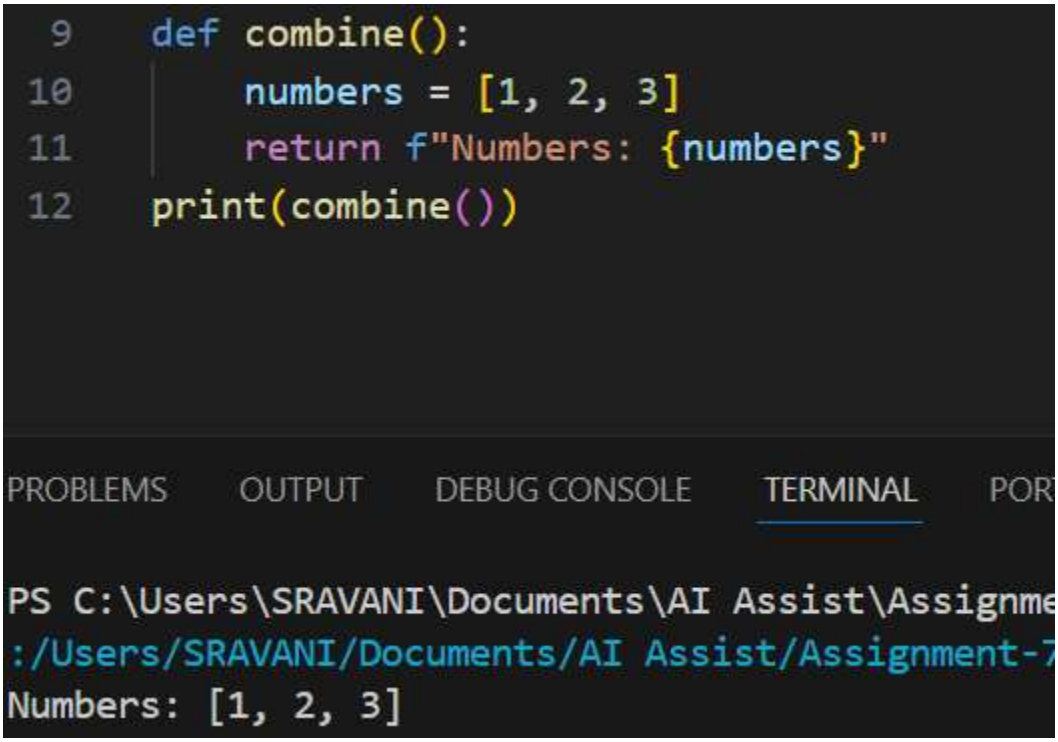
• Verify with 3 assert cases.

Expected Output:

• Corrected code

• Explanation

• Successful test validation



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

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\SRAVANI\Documents\AI Assist\Assignment-7.5> & C:/Users/SRAVANI/AppData/Loc
:/Users/SRAVANI/Documents/AI Assist/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.

```python
def compute():
    value = 0  # Assign a default value
    return value + 10
result = compute()
print(result)
# Validate using asserts
assert result == 10, "Test 1 failed"
assert isinstance(result, int), "Test 2 failed"
assert result > 0, "Test 3 failed"
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.