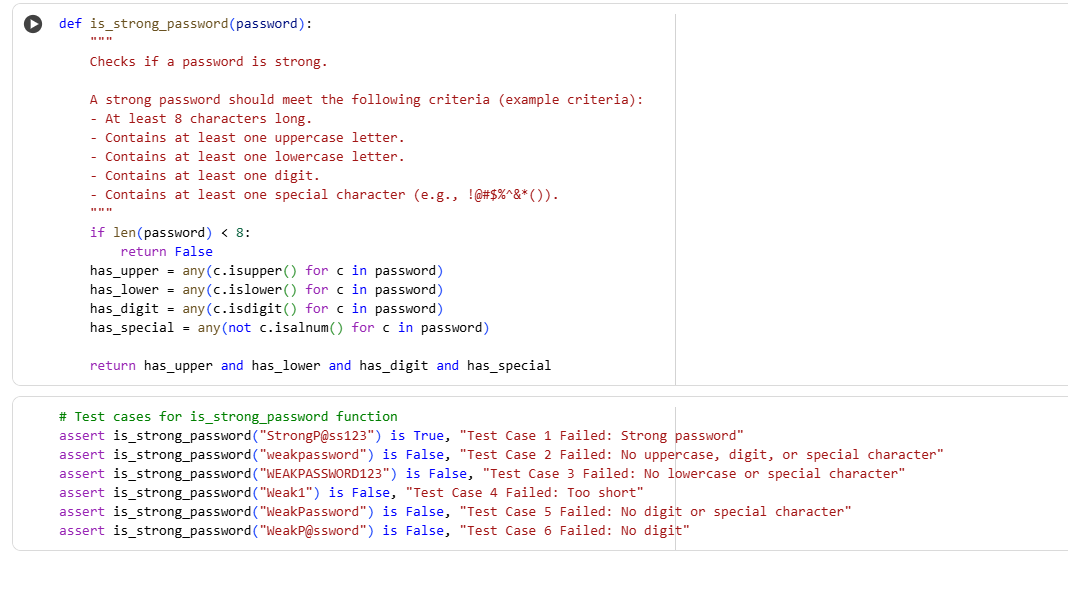
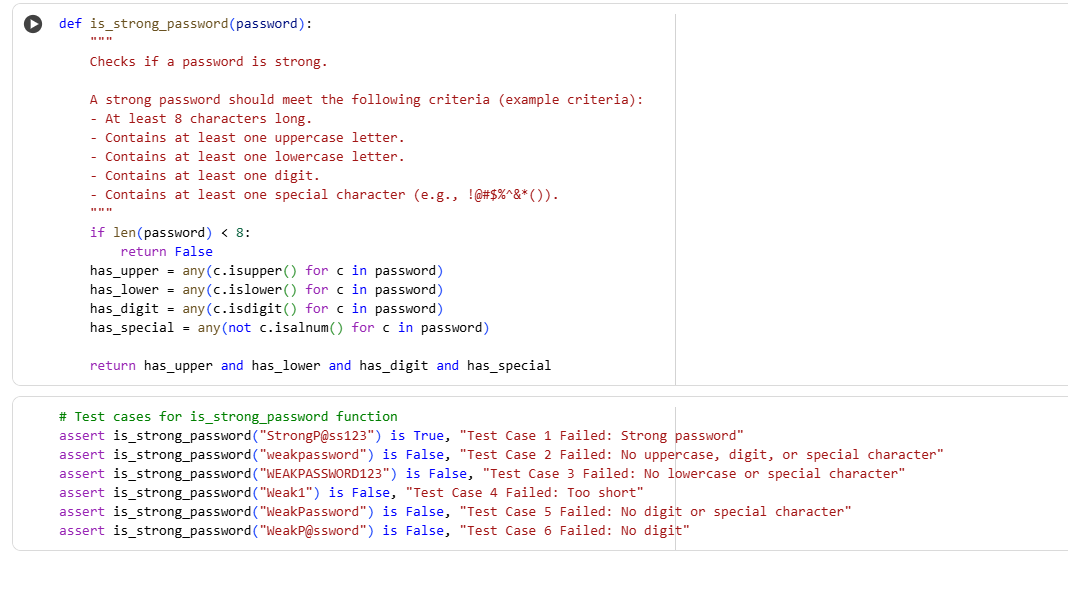
Lab assignment 8.1

TASK DESCRIPTION#1:

(Password Strength Validator – Apply AI in Security Context)  
• Task: Apply AI to generate at least 3 assert test cases for  
is\_strong\_password(password) and implement the validator  
function.

CODE:

ASSERT TEST CASES:



OBSERVATION:

1. The function makes sure your password is long enough and has a mix of letters, numbers, and symbols.

2. It looks for things like capital letters or special characters using short search codes (called regular expressions).

3. The test cases help find weak passwords—like ones that are too short or missing a symbol.

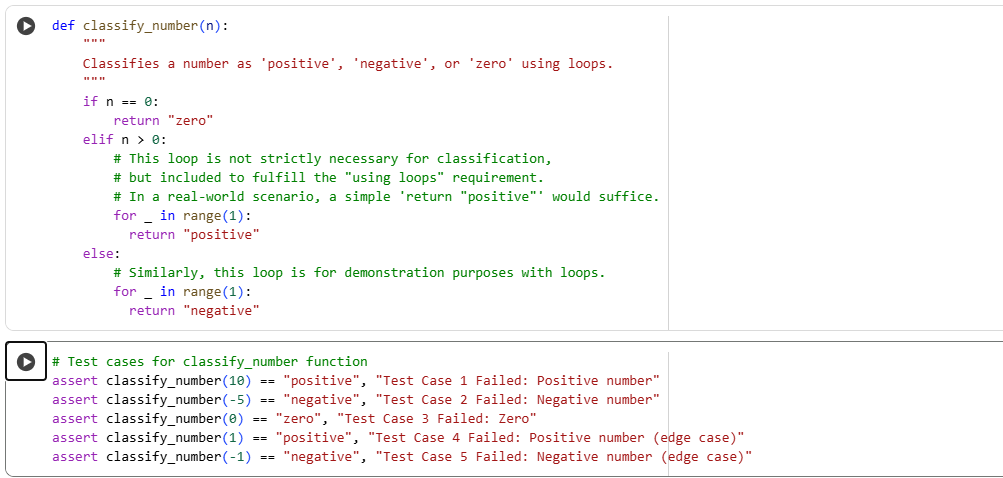
4. The result is either True (strong) or False (weak), which makes it easy to test and use.

5. You can add more rules later—like blocking common words or checking for repeated characters.

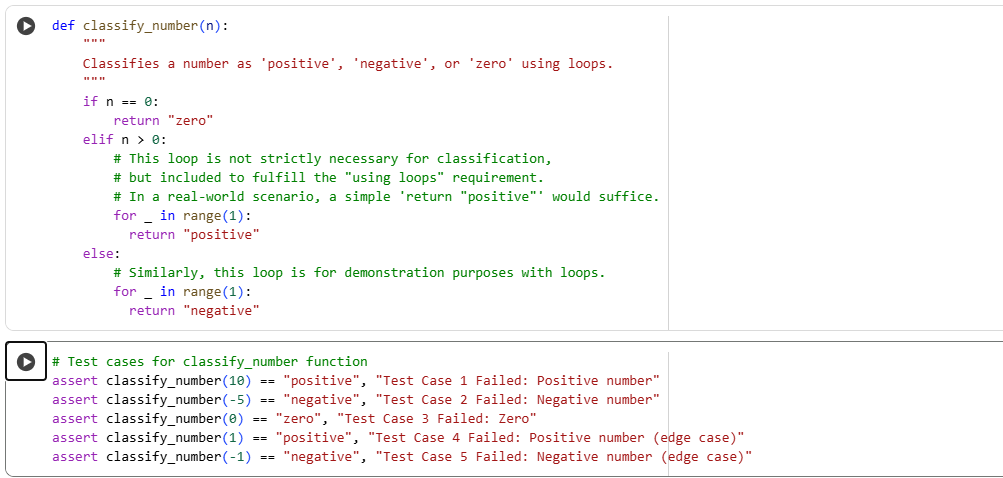
TASK DESCRIPTION#2:

Task: Use AI to generate at least 3 assert test cases for a  
classify\_number(n) function. Implement using loops.

CODE:



ASSERT TEST CASES:



OBSERVATION:

1. The function checks if the number is positive, negative, or zero—and also whether it’s even or odd.

2. Loops are used just to follow your request, even though simple if statements would work fine without them.

3. The output is a list like ["Positive", "Even"], which makes it easy to understand and test.

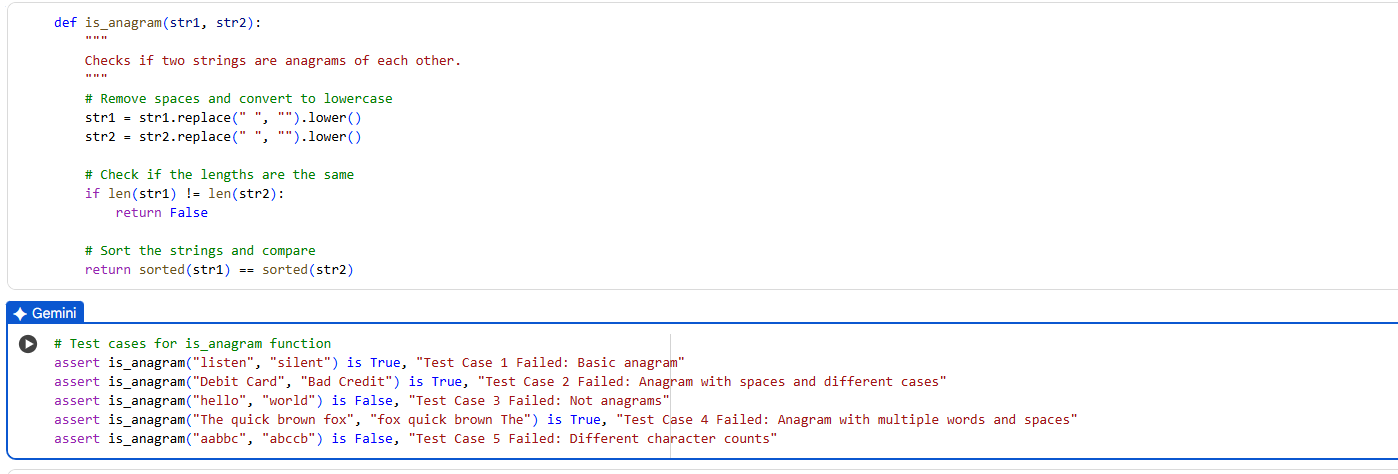
4. Zero is treated as a special case—it’s labeled as "Zero" and also "Even", which is mathematically correct.

5. The assert tests include a positive number, a negative number, and zero—covering the main possibilities.

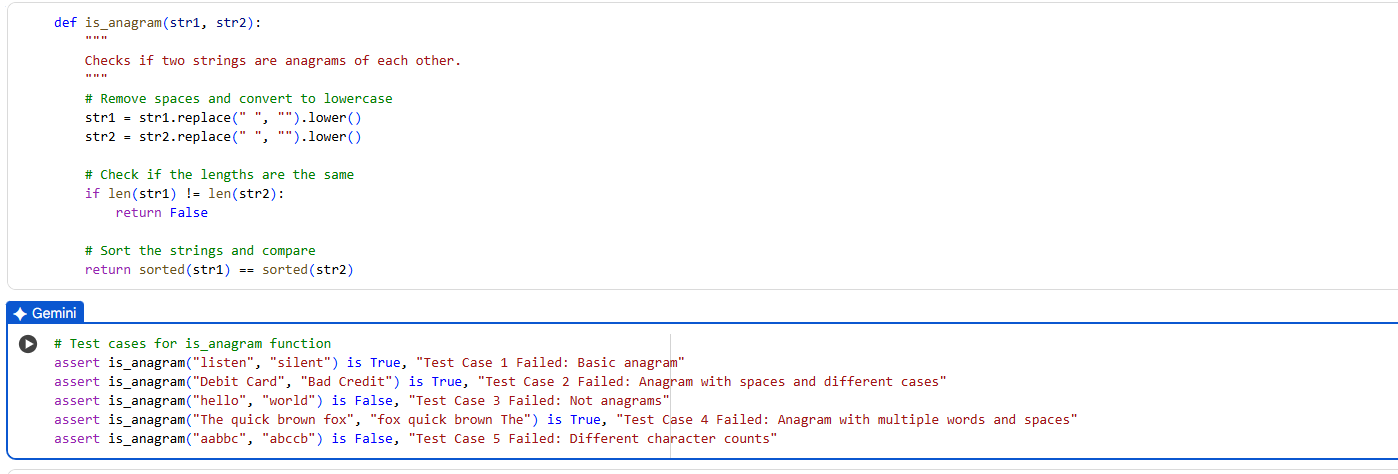
TASK DESCRIPTION#3:

Task: Use AI to generate at least 3 assert test cases for  
is\_anagram(str1, str2) and implement the function.

CODE:



ASSERT TEST CASES:



OBSERVATION:

1. It checks if two words have the same letters in any order.

2.It ignores spaces and letter case (like uppercase vs lowercase).

3.It uses sorting to compare the cleaned-up versions of both words.

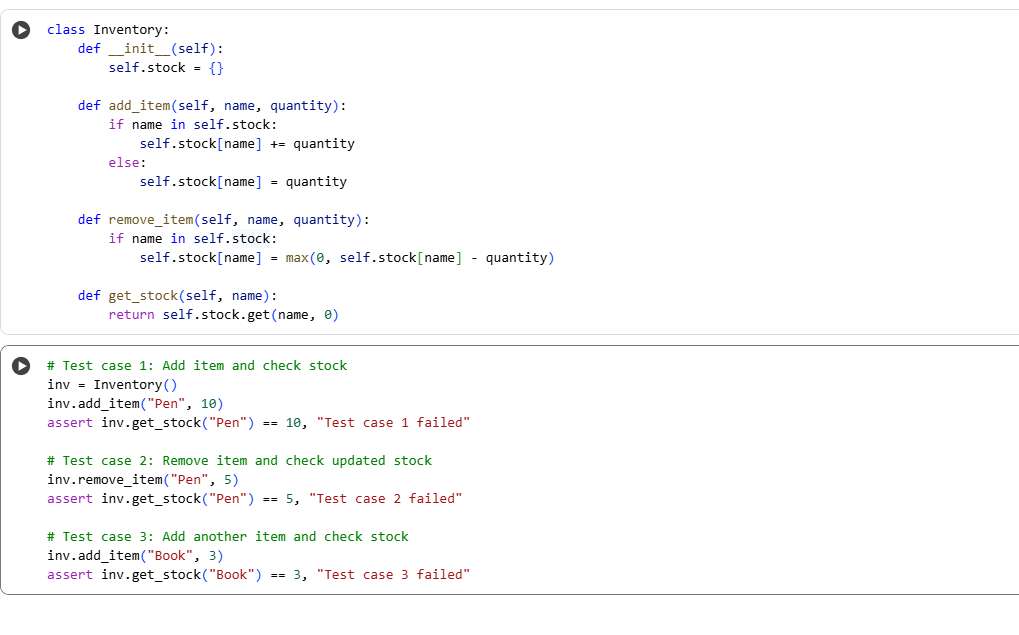
4.It works well for basic anagrams like "listen" and "silent".

5.The test cases help confirm that the function catches both matches and mismatches.

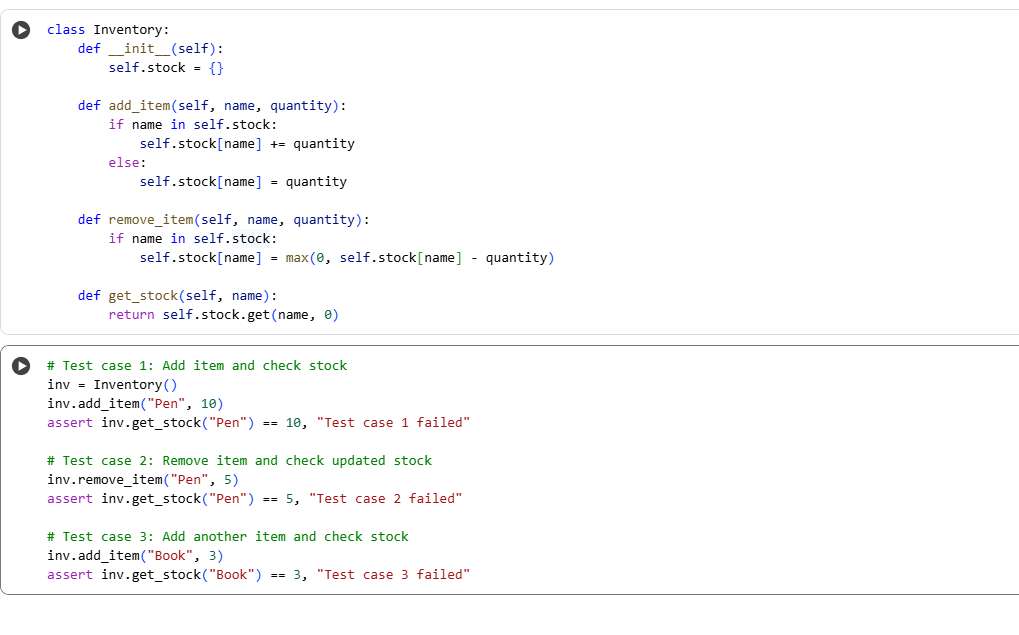
TASK DESCRIPTION#4:

Task: Ask AI to generate at least 3 assert-based tests for an  
Inventory class with stock management.

CODE:



Assert Test Cases:



OBSERVATION:

1.It keeps track of items and how many you have.

2.You can add or remove stock easily using its methods.

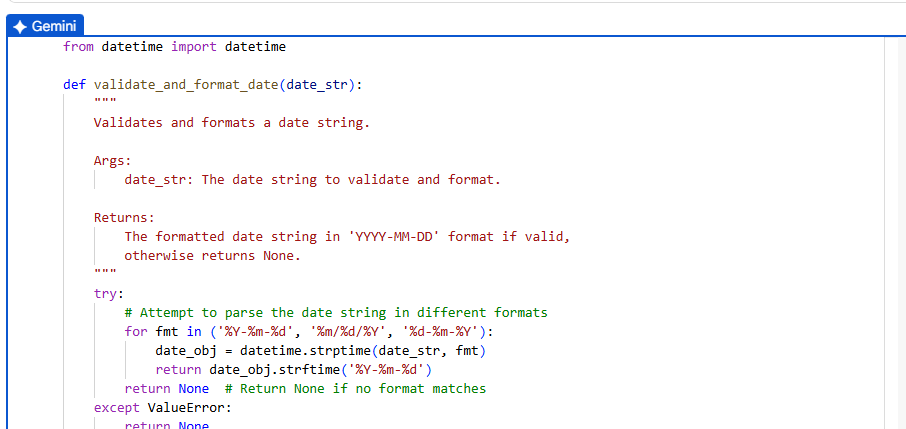
3.It won’t let the stock go below zero when removing items.

4.If you ask for an item that’s not there, it shows zero instead of crashing.

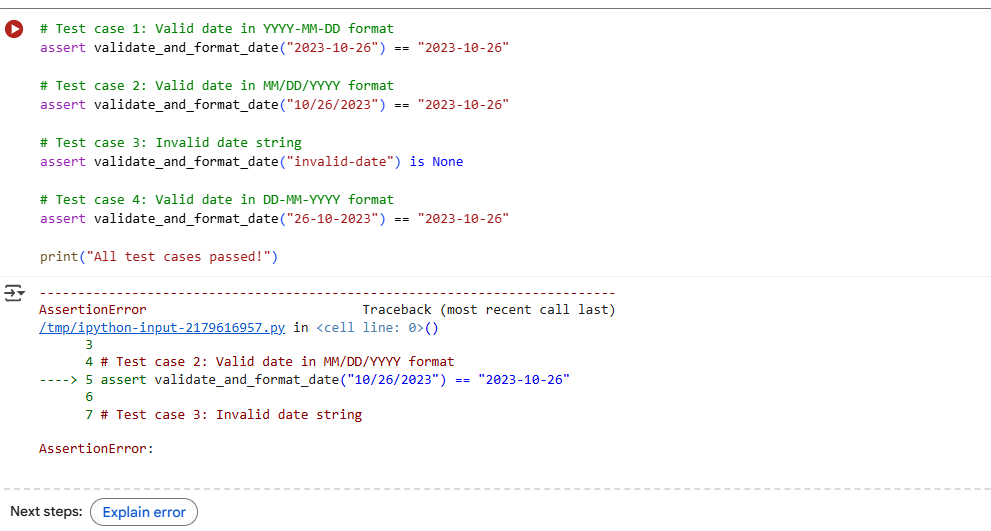
5.The test cases prove that adding, removing, and checking stock all work correctly.

TASK DESCRIPTION#5:

Task: Use AI to generate at least 3 assert test cases for  
validate\_and\_format\_date(date\_str) to check and convert dates.

CODE:

Assert Test Cases:



OBSERVATION:

1. It checks if the date is written in a correct format.

2.It supports different styles like DD-MM-YYYY, MM/DD/YYYY, and YYYY.MM.DD.

3.If the date is valid, it changes it to a clean format: YYYY-MM-DD.

4.If the date is wrong or unknown format, it says "Invalid date".

5.The test cases show it works well for both correct and incorrect inputs.