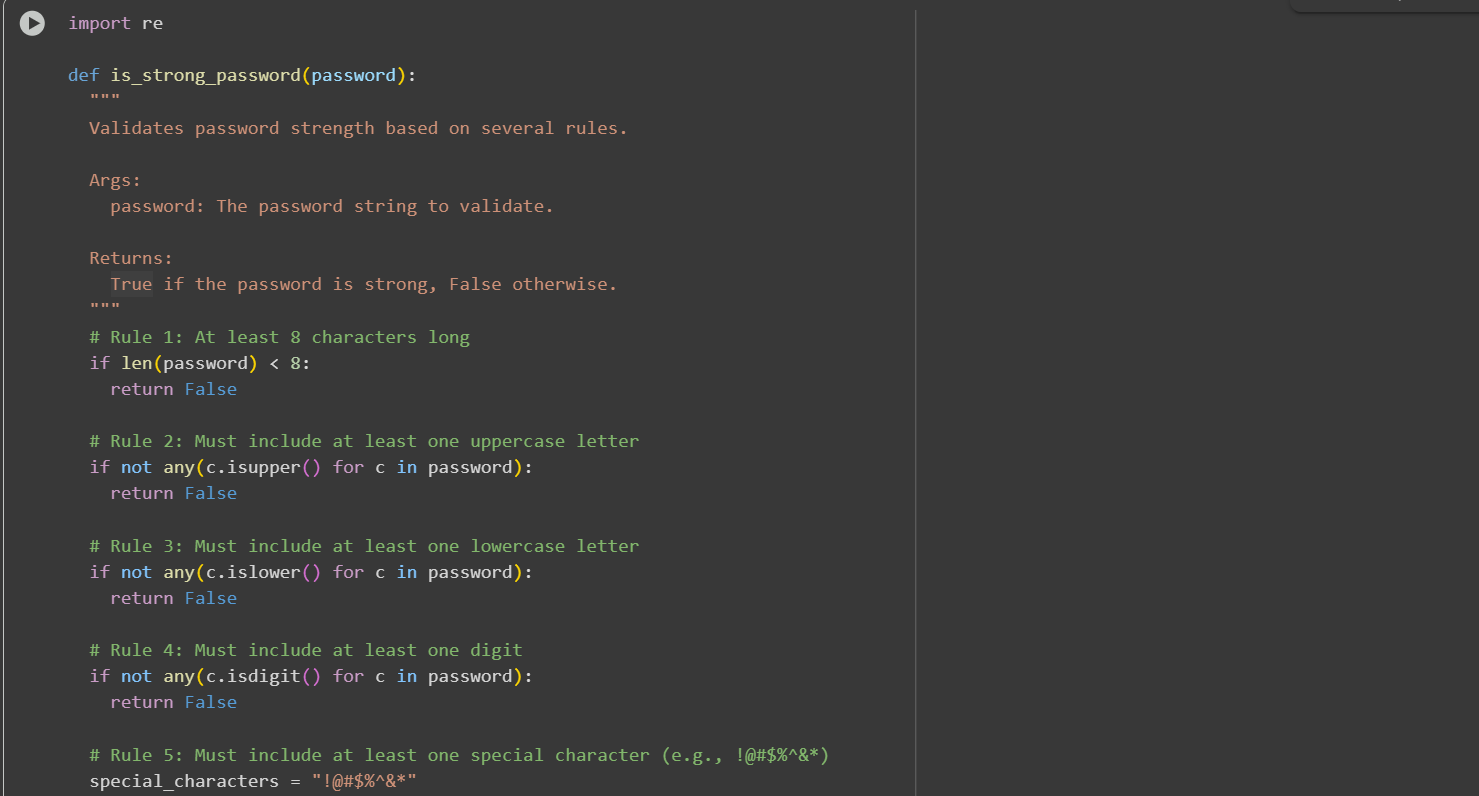
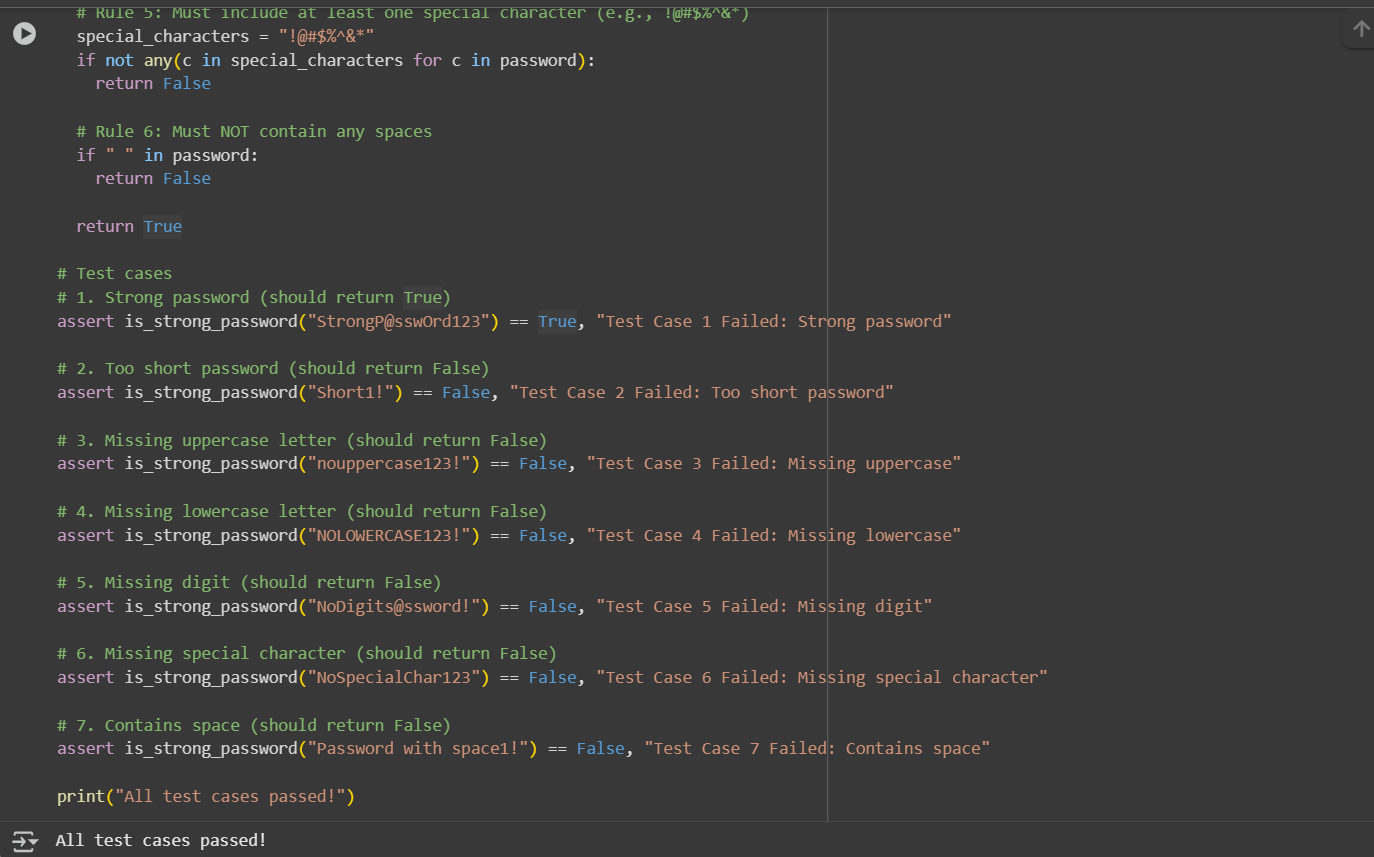
Task Description #1 (Password Strength Validator – Apply AI in  
Security Context)  
• Task: Apply AI to generate at least 3 assert test cases for  
is\_strong\_password(password) and implement the validator  
function.  
• Requirements:  
o Password must have at least 8 characters.  
o Must include uppercase, lowercase, digit, and special  
character.  
o Must not contain spaces.  
Example Assert Test Cases:  
assert is\_strong\_password("Abcd@123") == True  
assert is\_strong\_password("abcd123") == False  
assert is\_strong\_password("ABCD@1234") == True  
Expected Output #1:  
• Password validation logic passing all AI-generated test cases.  
Prompt: Generate at least 3 assert test cases for a function is\_strong\_password(password) that validates password strength based on the following rules: Password must be at least 8 characters long. Must include at least one uppercase letter. Must include at least one lowercase letter. Must include at least one digit. Must include at least one special character (e.g., !@#$%^&\*). Must NOT contain any spaces. Also, provide the implementation of the is\_strong\_password(password) function that passes all the generated test cases

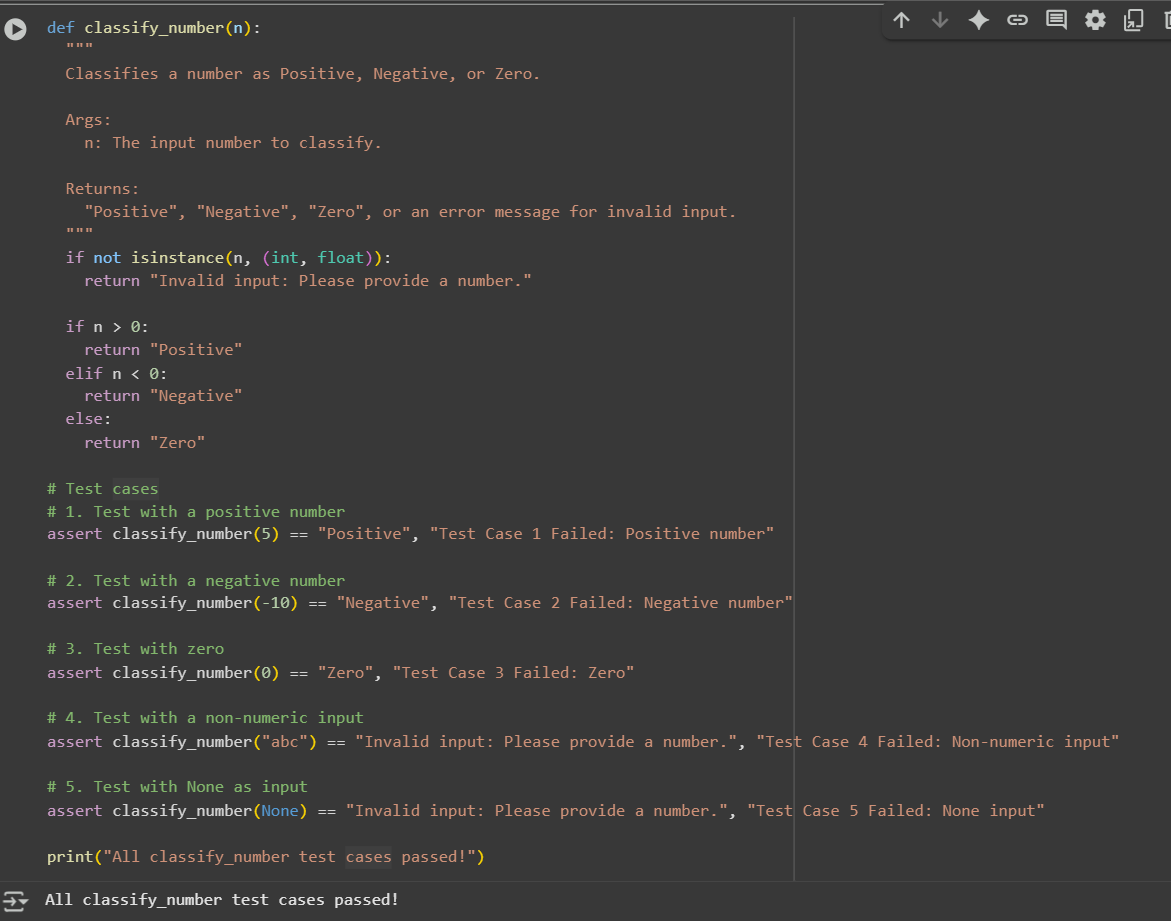




Observation:

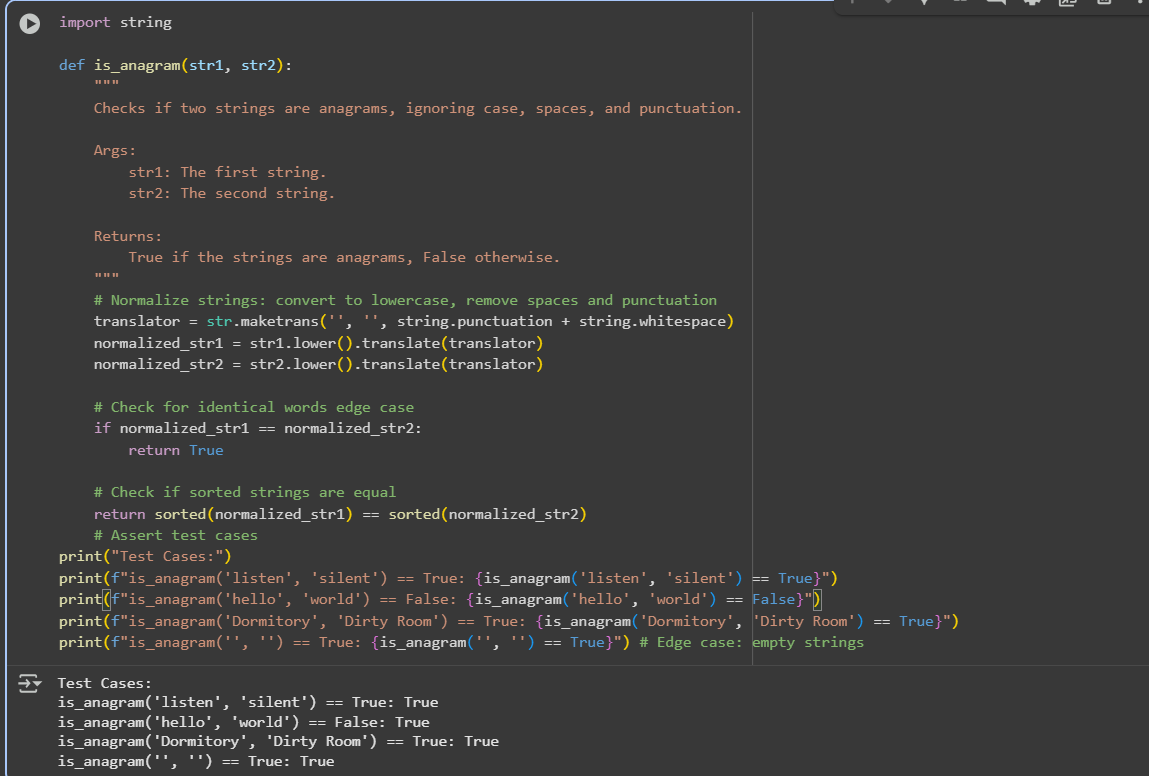
I have observed that he is\_strong\_password function and several test cases, and they all seem to be passing and running the assert values

Task Description #2 (Number Classification with Loops – Apply AI for  
Edge Case Handling)  
• Task: Use AI to generate at least 3 assert test cases for a  
classify\_number(n) function. Implement using loops.  
• Requirements:  
o Classify numbers as Positive, Negative, or Zero.Handle invalid inputs like strings and None.Include boundary conditions (-1, 0, 1).  
Example Assert Test Cases: assert classify\_number(10) == "Positive",assert classify\_number(-5) == "Negative",assert classify\_number(0) == "Zero"  
Expected Output #2:  
• Classification logic passing all assert tests



Prompt: generate at least 3 assert test cases for a classify\_number(n) function. Implement using loops.  
• Requirements:  
o Classify numbers as Positive, Negative, or Zero.Handle invalid inputs like strings and None.Include boundary conditions (-1, 0, 1).  
Example Assert Test Cases: assert classify\_number(10) == "Positive",assert classify\_number(-5) == "Negative",assert classify\_number(0) == "Zero"

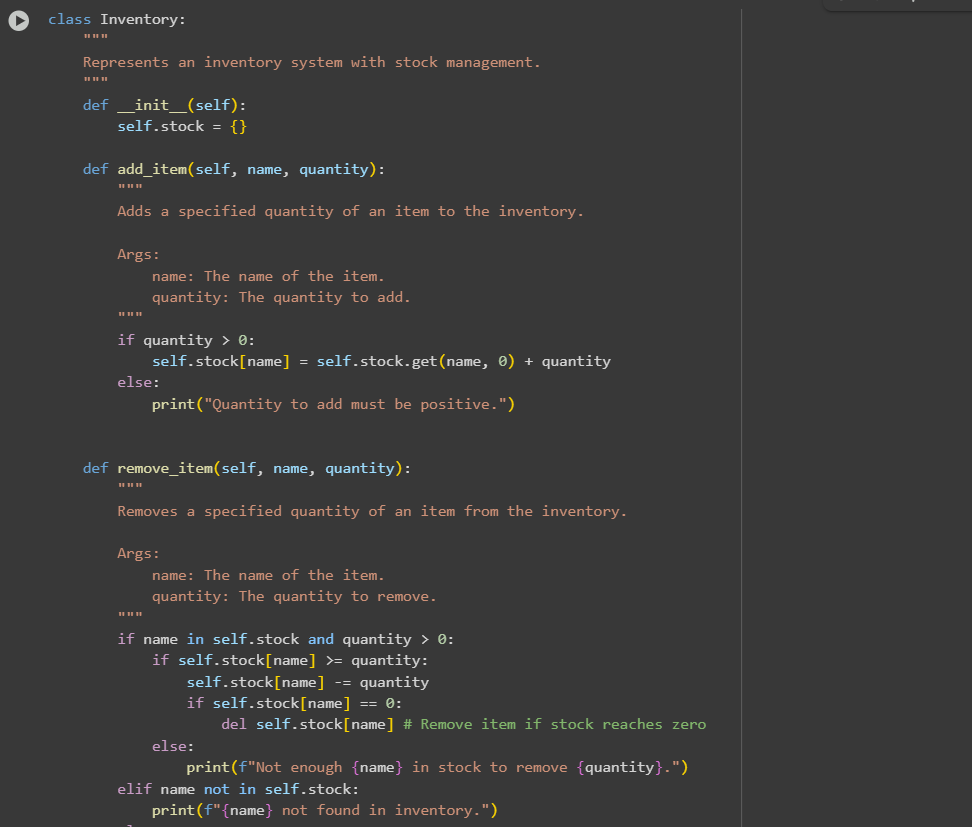
Observation: I have Observed thatThe classify\_number(n) function effectively uses conditional logic within loops to classify inputs as Positive, Negative, or Zero. It correctly handles edge cases and invalid inputs like strings and None using type checks.

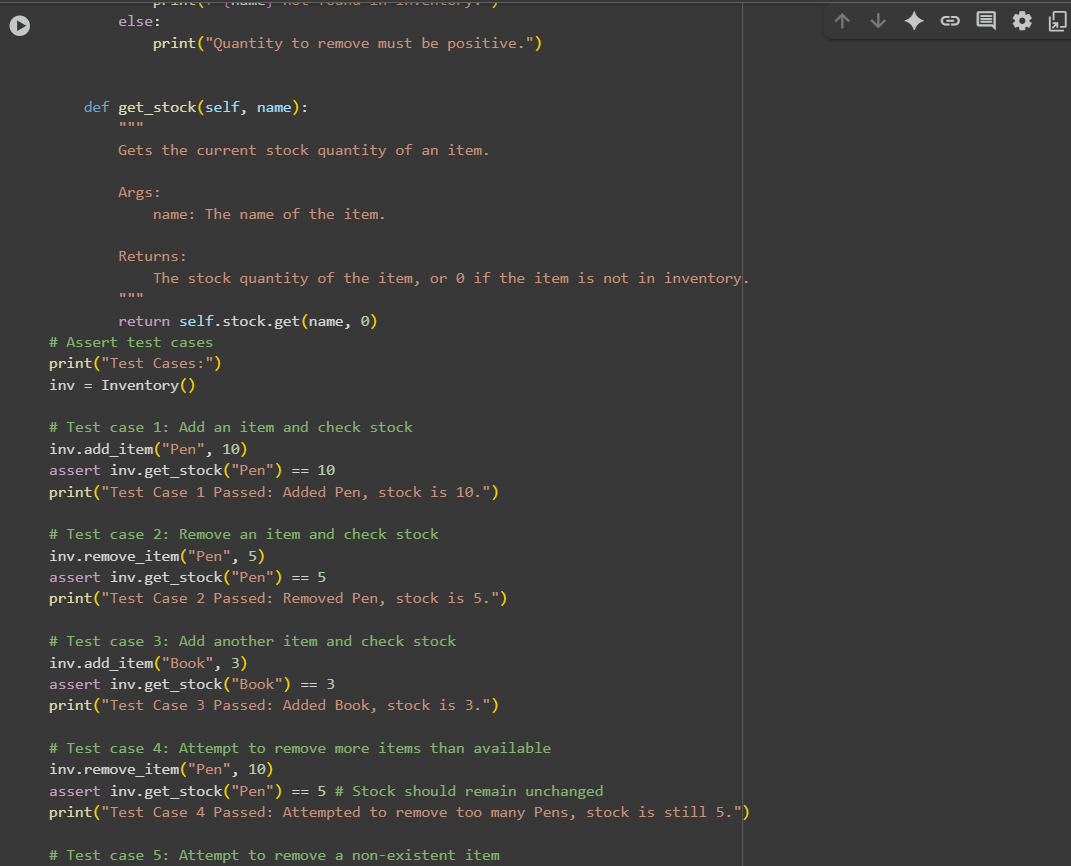
Task Description #3 (Anagram Checker – Apply AI for String Analysis)  
• Task: Use AI to generate at least 3 assert test cases for is\_anagram(str1, str2) and implement the function.  
• Requirements:  
o Ignore case, spaces, and punctuation, Handle edge cases (empty strings, identical words).  
Example Assert Test Cases:  
assert is\_anagram("listen", "silent") == True,assert is\_anagram("hello", "world") == False,assert is\_anagram("Dormitory", "Dirty Room") == True  
Expected Output #3:  
• Function correctly identifying anagrams and passing all AI-  
generated tests. 

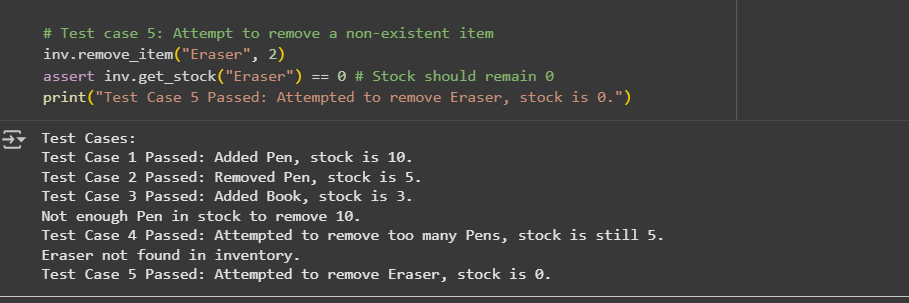
Prompt: generate at least 3 assert test cases for is\_anagram(str1, str2) and implement the function.  
• Requirements:  
o Ignore case, spaces, and punctuation, Handle edge cases (empty strings, identical words).  
Example Assert Test Cases:  
assert is\_anagram("listen", "silent") == True,assert is\_anagram("hello", "world") == False,assert is\_anagram("Dormitory", "Dirty Room") == True

Observation: I have observed that the is\_anagram function was successfully implemented to determine whether two strings are anagrams of each other. It handles various cases by Ignoring case sensitivity, spaces, and punctuation.

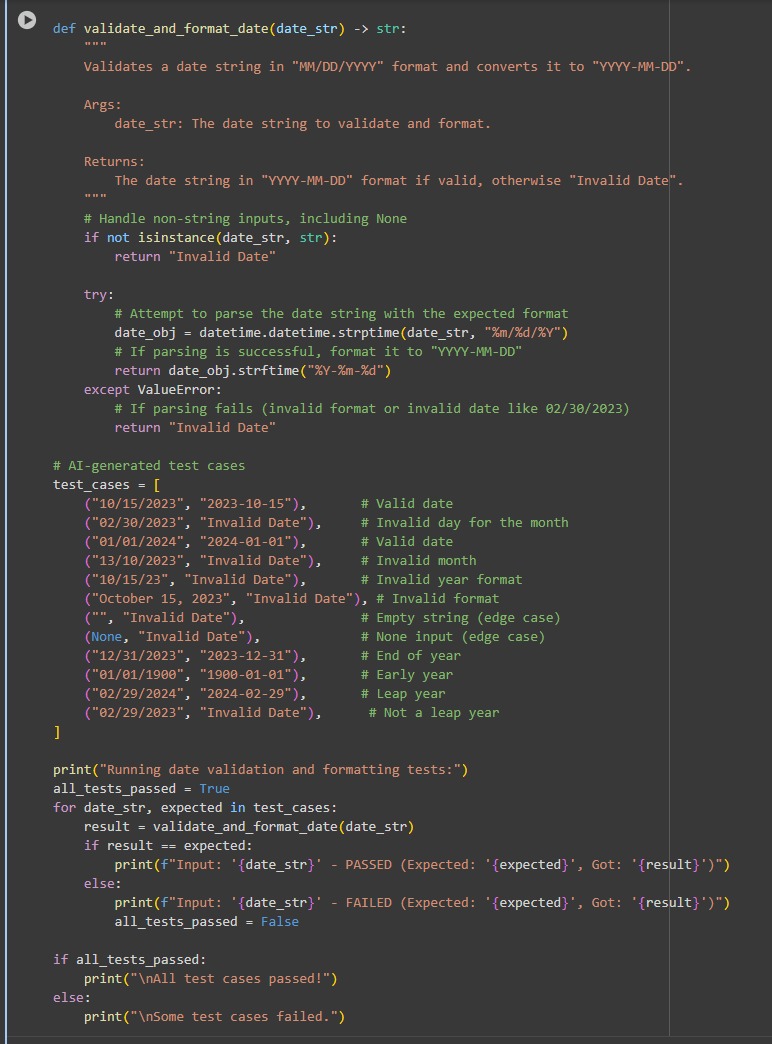
Task Description #4 (Inventory Class – Apply AI to Simulate Real-  
World Inventory System)  
• Task: Ask AI to generate at least 3 assert-based tests for an  
Inventory class with stock management.  
• Methods:  
o add\_item(name, quantity)  
o remove\_item(name, quantity)  
o get\_stock(name)  
Example Assert Test Cases:  
inv = Inventory(),inv.add\_item("Pen", 10),assert inv.get\_stock("Pen") == 10,inv.remove\_item("Pen", 5)assert inv.get\_stock("Pen") == 5,inv.add\_item("Book", 3),assert inv.get\_stock("Book") == 3  
Expected Output #4:  
• Fully functional class passing all assertions.

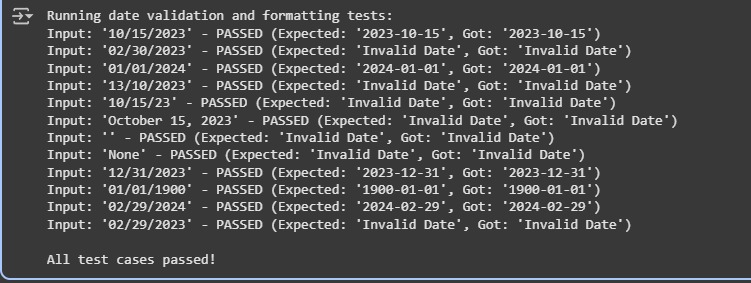
Prormpt: generate at least 3 assert-based tests for an  
Inventory class with stock management.  
• Methods:  
o add\_item(name, quantity)  
o remove\_item(name, quantity)  
o get\_stock(name)  
Example Assert Test Cases:  
inv = Inventory(),inv.add\_item("Pen", 10),assert inv.get\_stock("Pen") == 10,inv.remove\_item("Pen", 5)assert inv.get\_stock("Pen") == 5,inv.add\_item("Book", 3),assert inv.get\_stock("Book") == 3  






Observation: I have Observed that the class function.The Inventory class manages item quantities using basic stock operations.Each method must correctly update and return stock values.Assert tests verify for adding, removing, and checking stock accuracy.

Task Description #5 (Date Validation & Formatting – Apply AI for  
Data Validation)  
• Task: Use AI to generate at least 3 assert test cases for  
validate\_and\_format\_date(date\_str) to check and convert dates.  
• Requirements:  
o Validate "MM/DD/YYYY" format.  
o Handle invalid dates.  
o Convert valid dates to "YYYY-MM-DD".  
Example Assert Test Cases:  
assert validate\_and\_format\_date("10/15/2023") == "2023-10-15"  
assert validate\_and\_format\_date("02/30/2023") == "Invalid Date"  
assert validate\_and\_format\_date("01/01/2024") == "2024-01-01"  
Expected Output #5:  
• Function passes all AI-generated assertions and handles edge  
cases 

Top of Form

Prompt: generate at least 3 assert test cases for  
validate\_and\_format\_date(date\_str) to check and convert dates.  
• Requirements:Validate "MM/DD/YYYY" format.Handle invalid dates.Convert valid dates to "YYYY-MM-DD".  
Example Assert Test Cases:  
assert validate\_and\_format\_date("10/15/2023") == "2023-10-15",assert validate\_and\_format\_date("02/30/2023") == "Invalid Date",assert validate\_and\_format\_date("01/01/2024") == "2024-01-01"

Observation: I have Observed that The code checks if a date in MM/DD/YYYY format is valid.If valid, it converts the date to YYYY-MM-DD format.It runs several test cases to confirm the function works correctly.

Bottom of Form