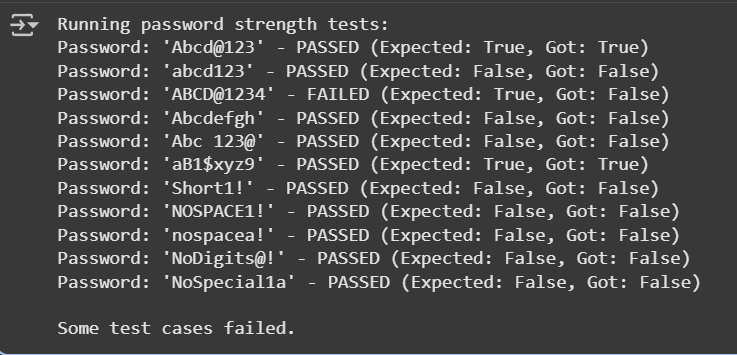
***Assignment 8.1***

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.  
• 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 a code to check all the Password validations and testing all the Cases of the Password

**Code and Output**:



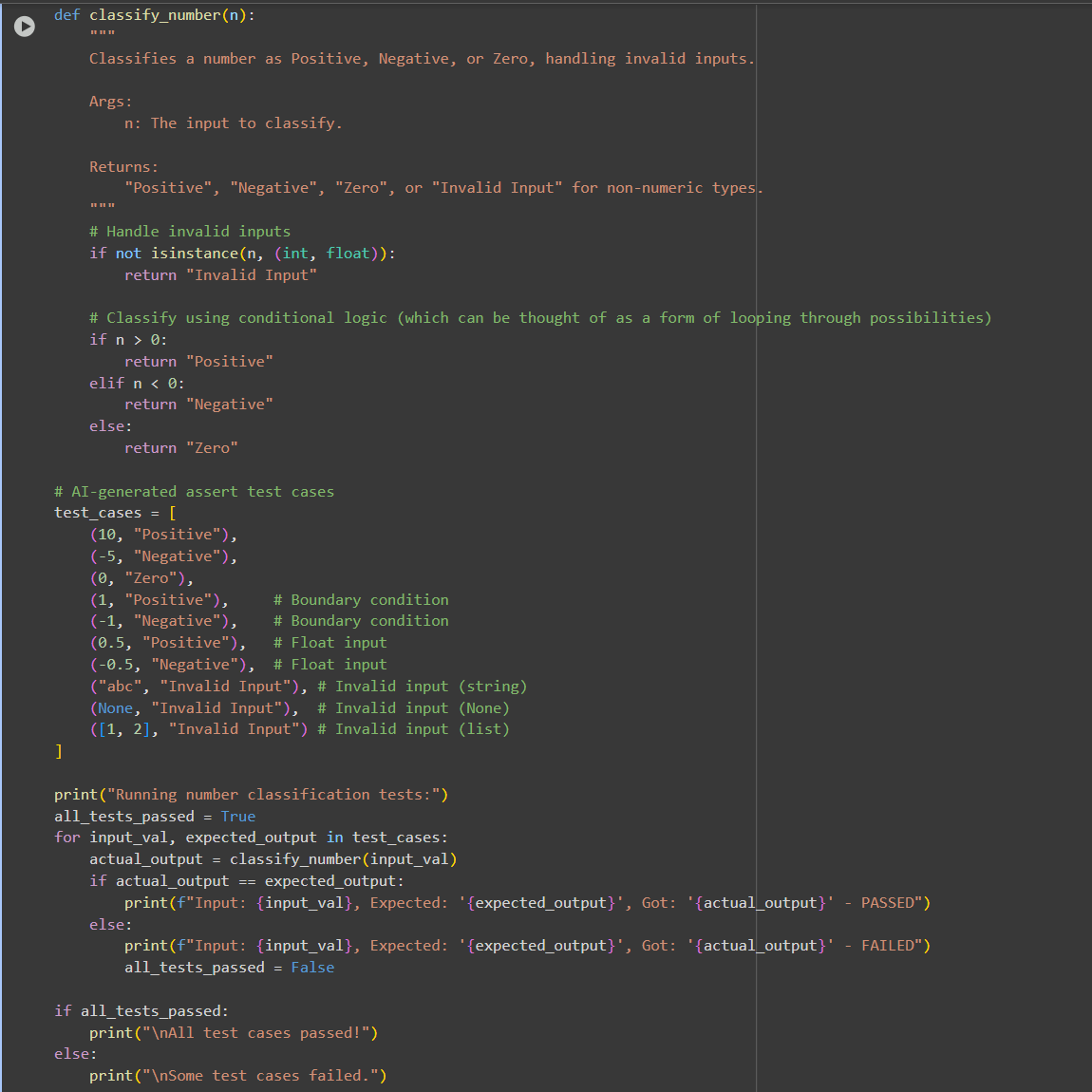
**Observation**: In this code we can Observe that the code is testing all the Password cases and giving the Output that it passed all the validations or not

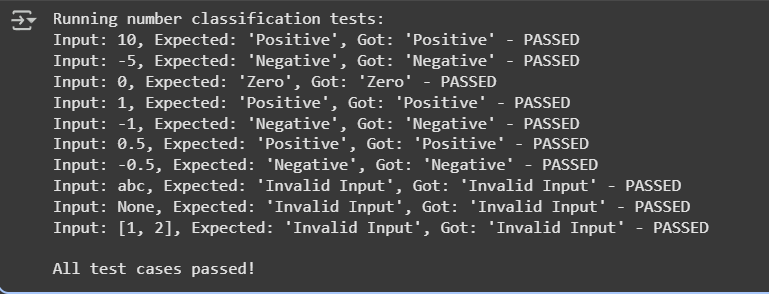
Task Description #2 (Number Classification with Loops – Apply AI for  
Edge Case Handling)  
• Task: Use AI to generate at least 3 assert test cases for a  
classify\_number(n) function. Implement using loops.  
• Requirements:  
o Classify numbers as Positive, Negative, or Zero.  
o Handle invalid inputs like strings and None.

o 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 a code to check whether the input number is a positive number, Negative Number , Invalid Number

**Code and Output:**

****



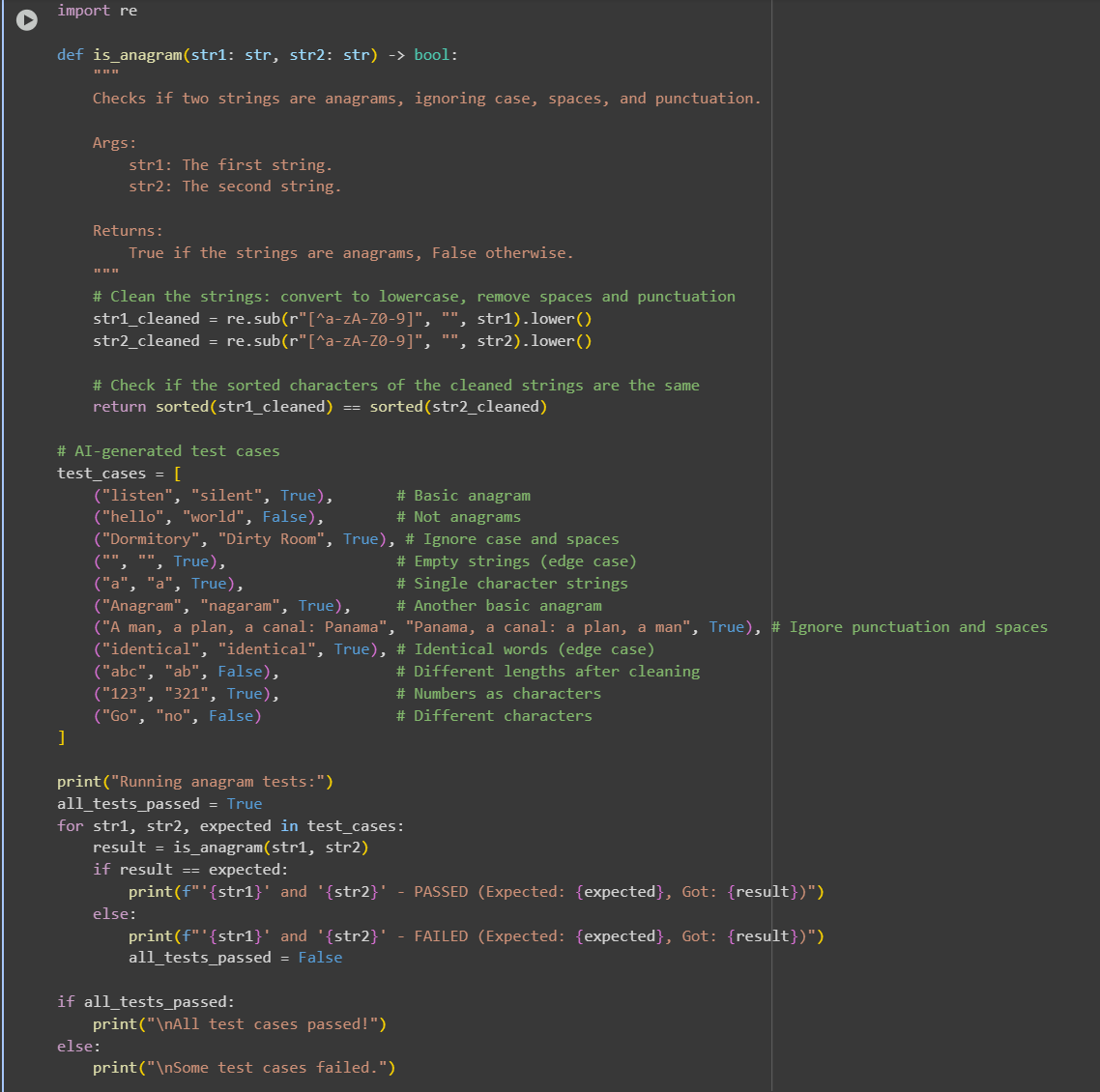
**Observation**:

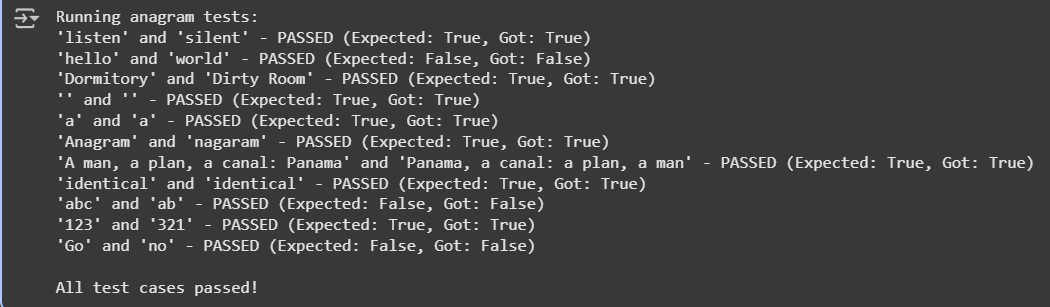
In this code we can Observe that the code is giving the Output of the Number is positive , negative or invalid input.

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.  
o 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 a code to check whether the input is anagram or not .

**Code and Output:**





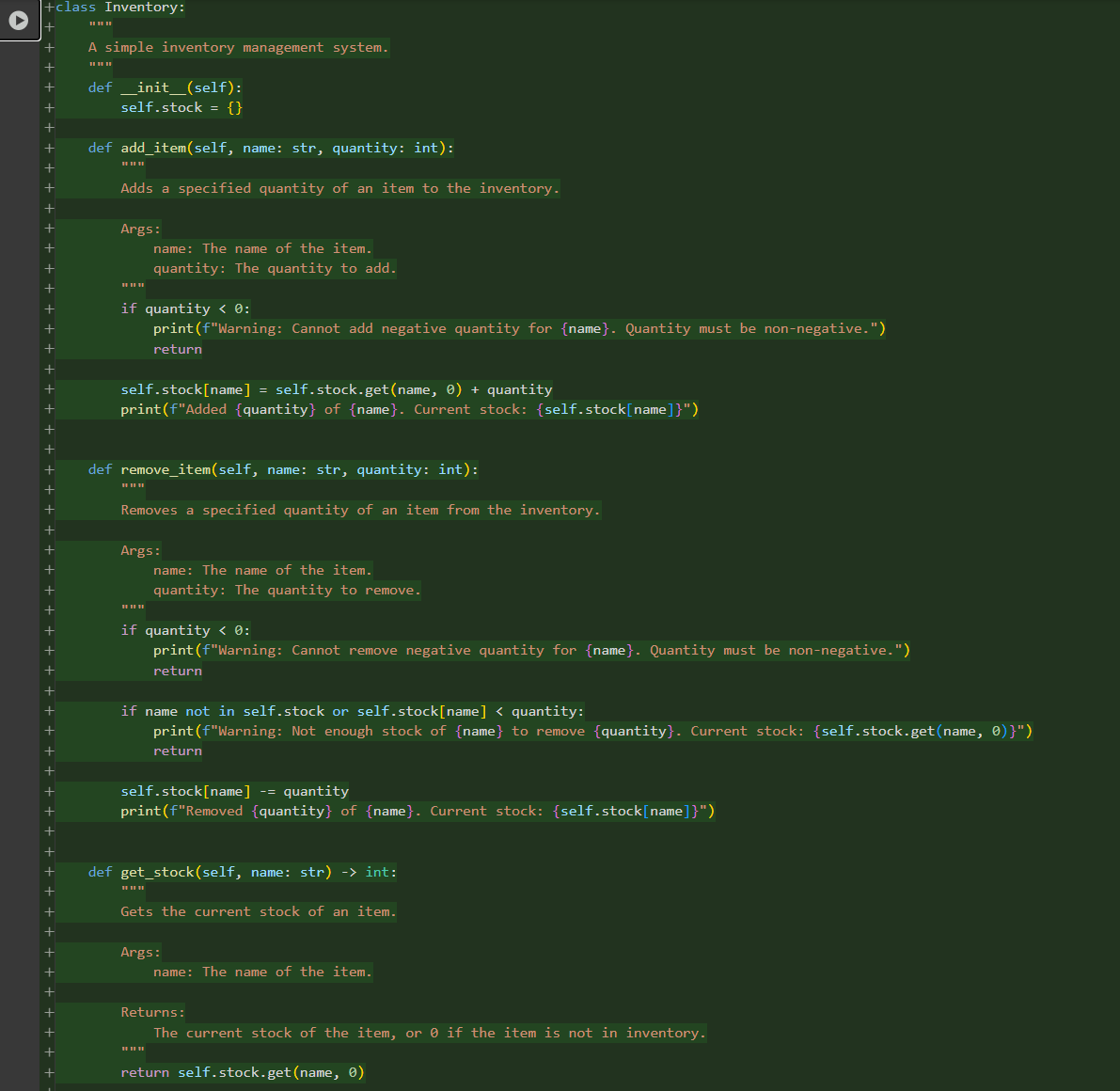
**Observation:**

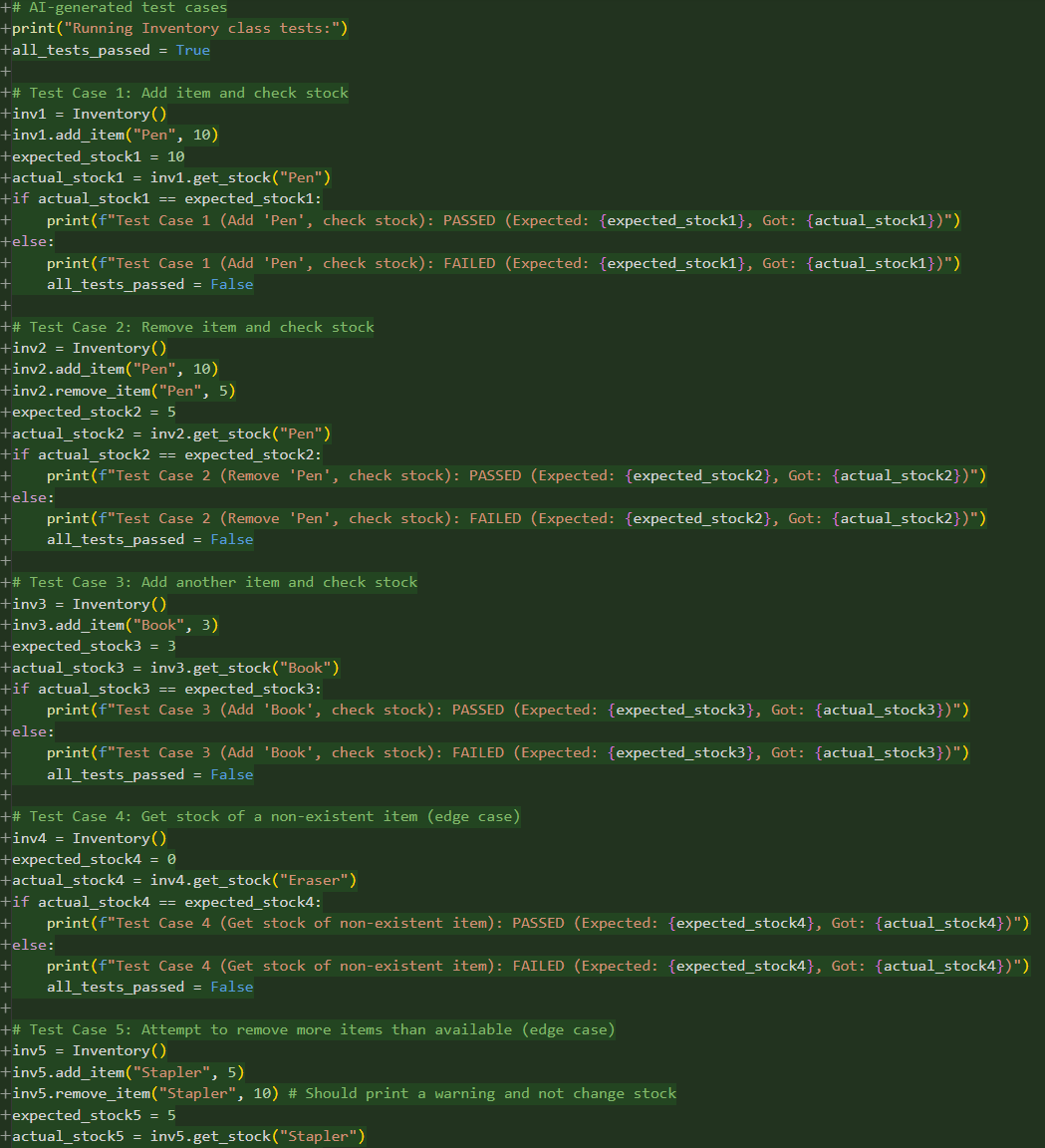
In this code We can see that the code is checking if the word is an anagram or not

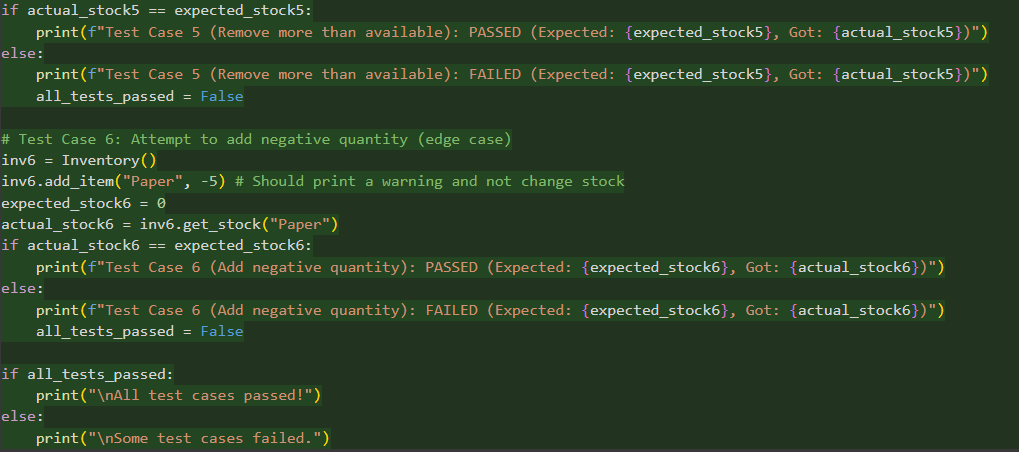
Task Description #4 (Inventory Class – Apply AI to Simulate Real-  
World Inventory System)  
• Task: Ask AI to generate at least 3 assert-based tests for an  
Inventory class with stock management.  
• Methods:  
o add\_item(name, quantity)  
o remove\_item(name, quantity)  
o get\_stock(name)  
Example 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.

**Prompt:** Generate a code to do this tasks

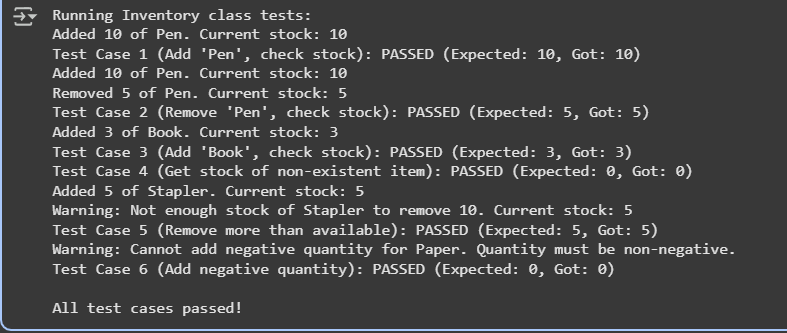
**Code and Output:**

****

****

****

**Output;**

****

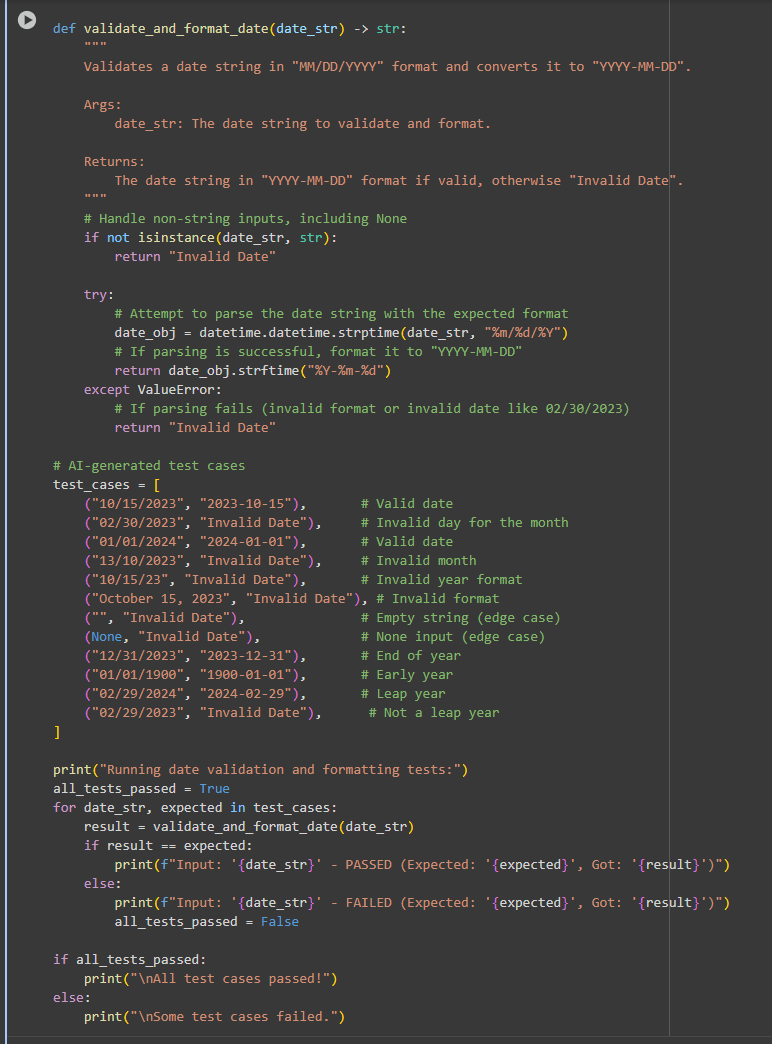
**Observation:**

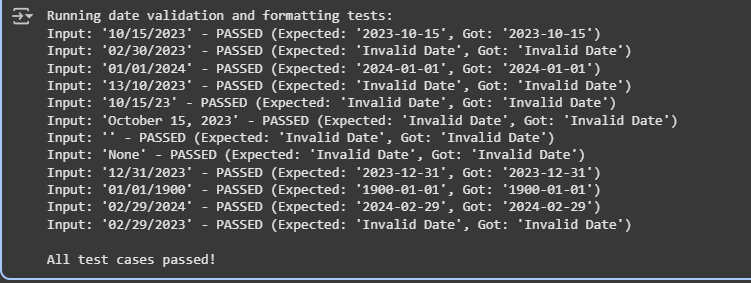
In this code we can Observe that the code we are adding pens , removing pens and also we are giving the warning that the pens cannot go into negative values.

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

Prompt:Generate a code for checking the validation of date using python in simple code

Code and Output:





Observation:

In this code we can observe that the code is checking for the date that is inputted is correct or not.