**ASSIGNMENT : 8.1**

**BATCH : 12**

**HALL TICKE:2403A51292**

**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:**

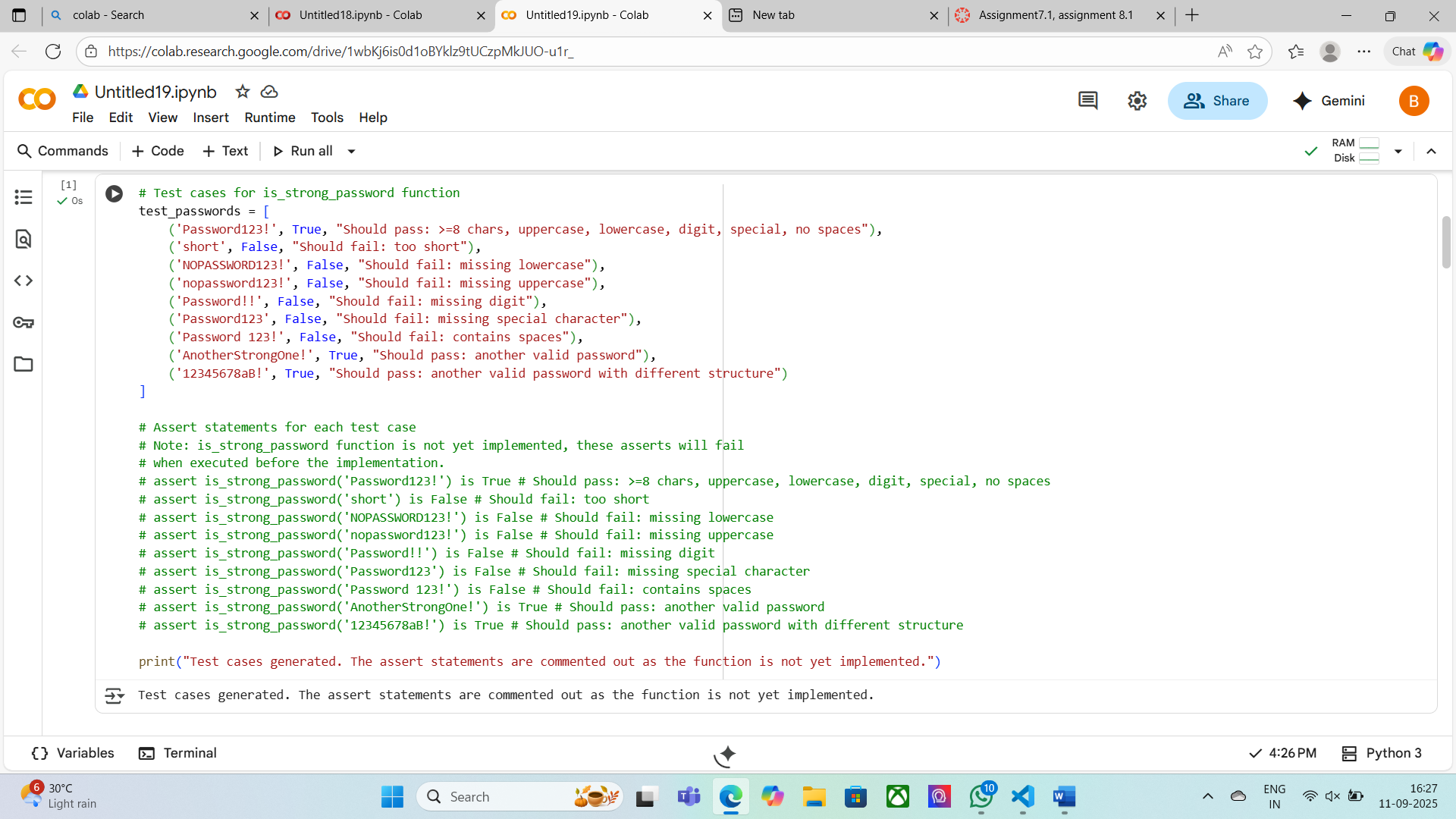
Implement the function [is\_strong\_password(password)](vscode-file://vscode-app/c:/Users/tejas/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) to validate password strength with the following requirements:

* At least 8 characters long
* Includes at least one uppercase letter, one lowercase letter, one digit, and one special character
* Must not contain spaces

1. Use AI to generate at least 3 assert test cases for the function.
2. Implement the password validator function.
3. Ensure the function passes all AI-generated test cases.

* Must not contain spaces

1. Use AI to generate at least 3 assert test cases for the function.
2. Implement the password validator function.
3. Ensure the function passes all AI-generated test cases.



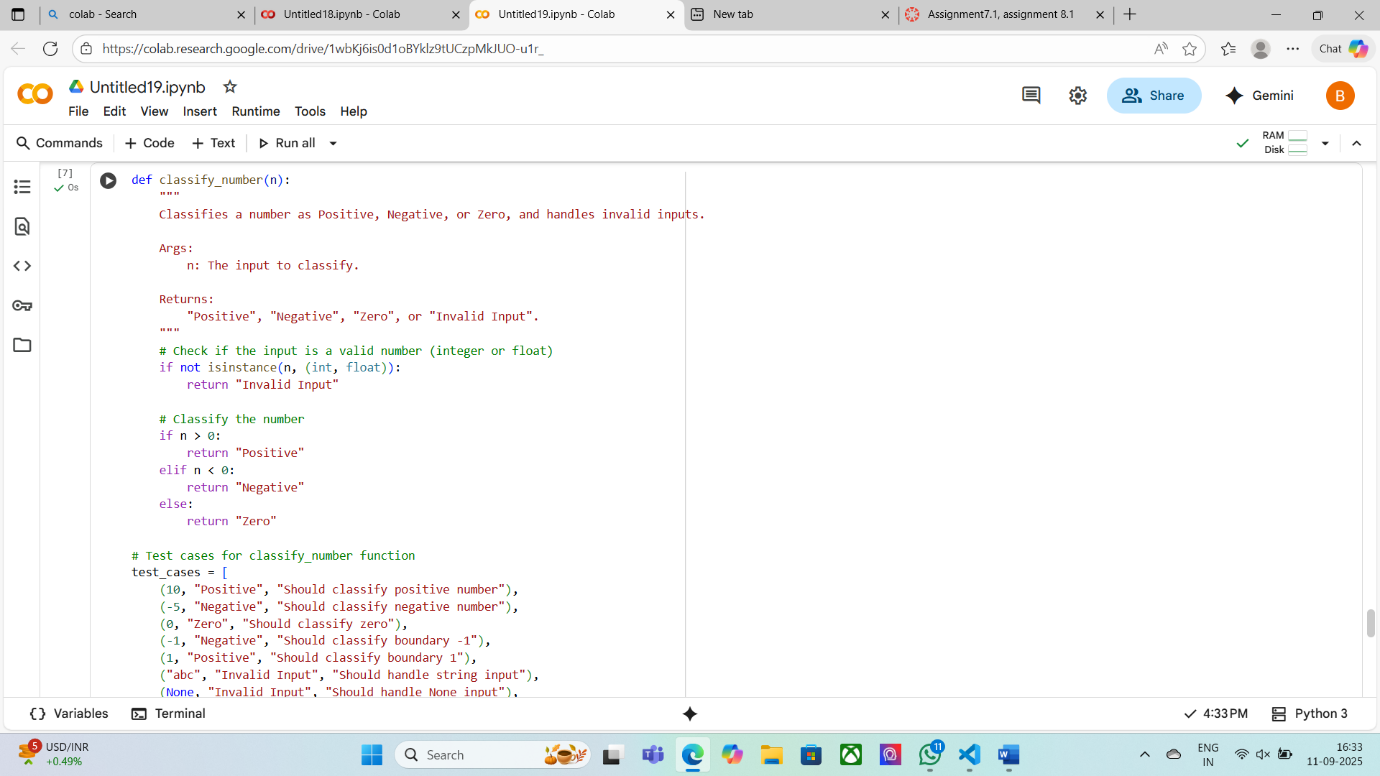
**Task Description 2 :**

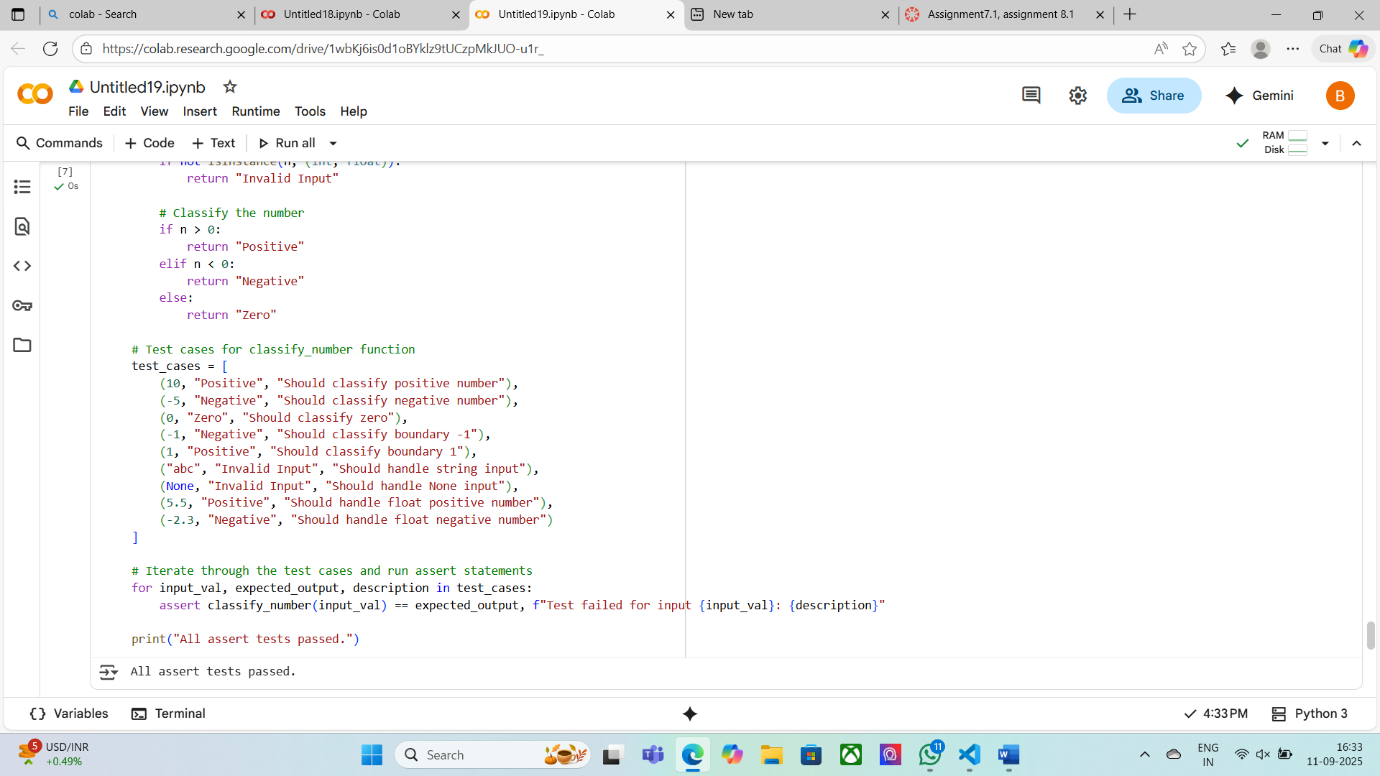
(Number Classification with Loops – Apply AI for  
Edge Case Handling)  
• Task: Use AI to generate at least 3 assert test cases for a  
classify\_number(n) function. Implement using loops.  
• Requirements:  
o Classify numbers as Positive, Negative, or Zero.  
o Handle invalid inputs like strings and 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:**  
Implement the function classify\_number(n) to classify numbers as "Positive", "Negative", or "Zero".

* The function should handle invalid inputs (like strings and None) and include boundary conditions (-1, 0, 1).
* Use loops if needed for input validation or processing.
* Use AI to generate at least 3 assert test cases.



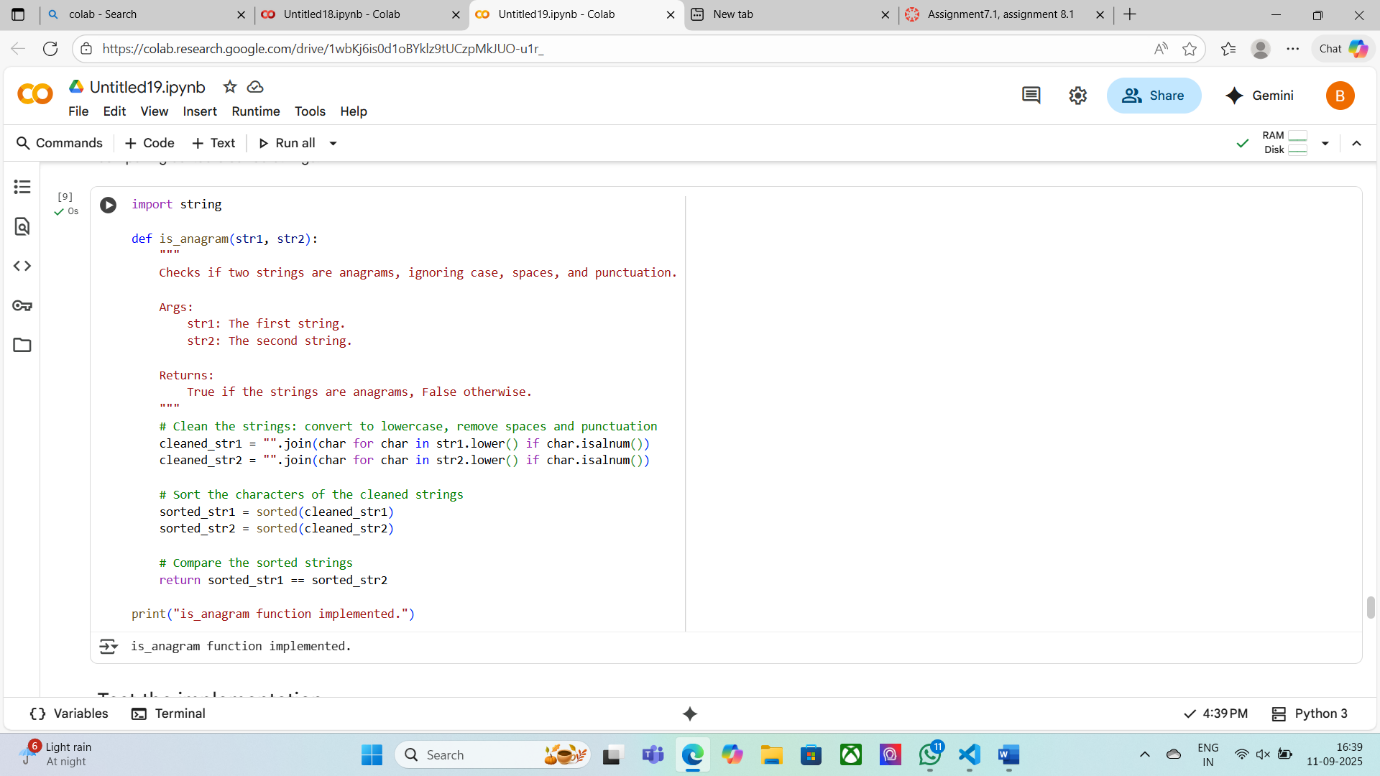


**Task Description 3 :**

(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:**  
Implement the function is\_anagram(str1, str2) to check if two strings are anagrams.

* Ignore case, spaces, and punctuation.
* Handle edge cases such as empty strings and identical words.
* Use AI to generate at least 3 assert test cases.



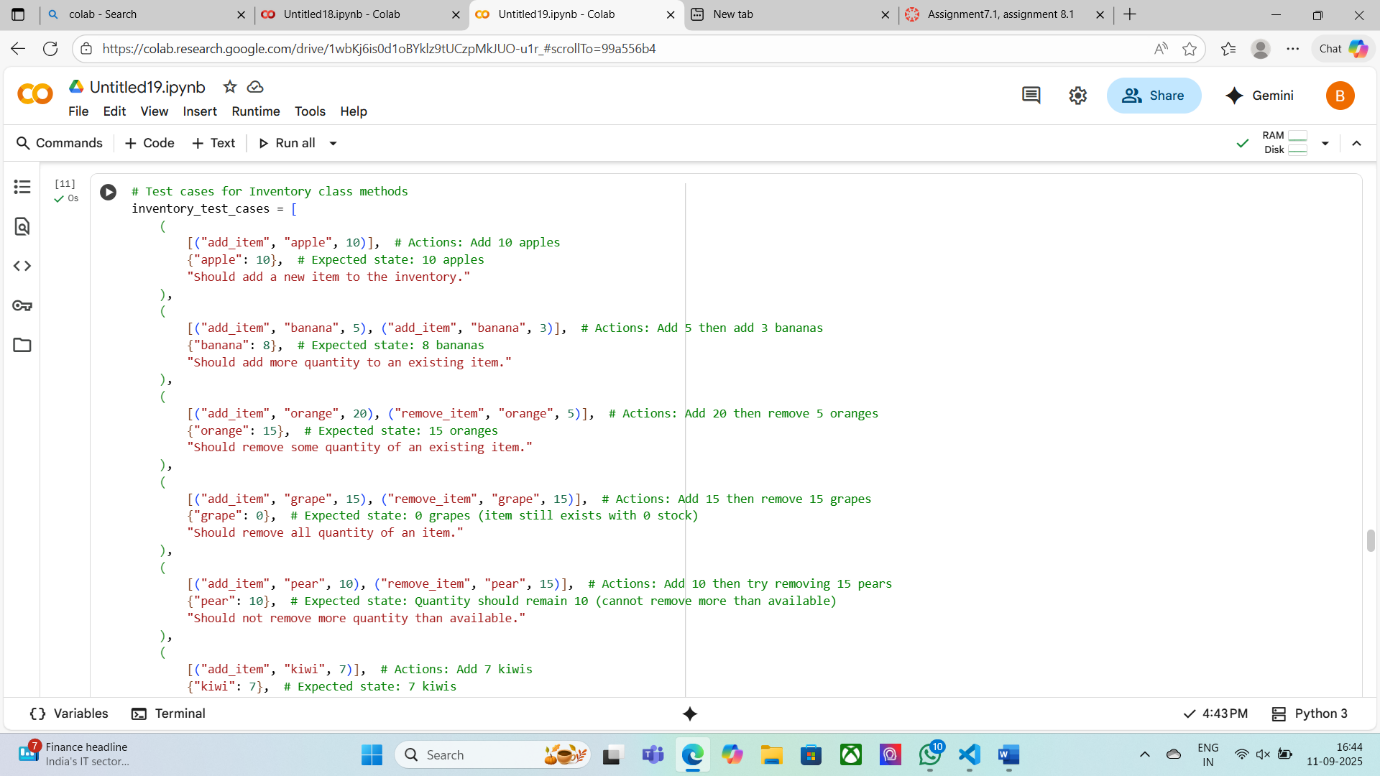
**Task Description 4 :**

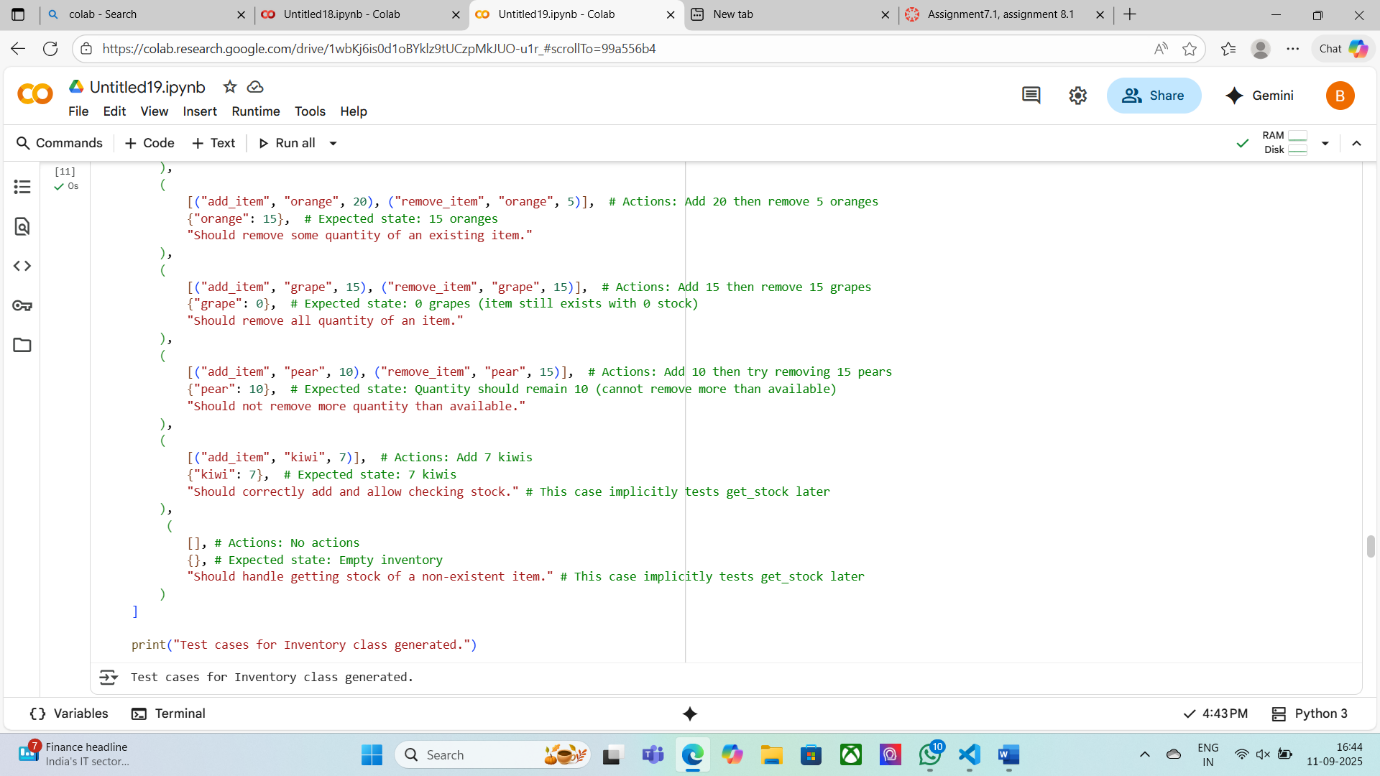
(Inventory Class – Apply AI to Simulate Real-  
World Inventory System)  
• Task: Ask AI to generate at least 3 assert-based tests for an  
Inventory class with stock management.  
• Methods:  
o add\_item(name, quantity)  
o remove\_item(name, quantity)  
o get\_stock(name)  
Example Assert Test Cases: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

Implement an Inventory class with methods for stock management:

* add\_item(name, quantity)
* remove\_item(name, quantity)
* get\_stock(name)
* Use AI to generate at least 3 assert-based tests for the class.





**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:**  
Implement the function validate\_and\_format\_date(date\_str) to:

* Validate dates in "MM/DD/YYYY" format.
* Handle invalid dates.
* Convert valid dates to "YYYY-MM-DD" format.
* Use AI to generate at least 3 assert test cases.

