**ASSIGNMENT-8.1**

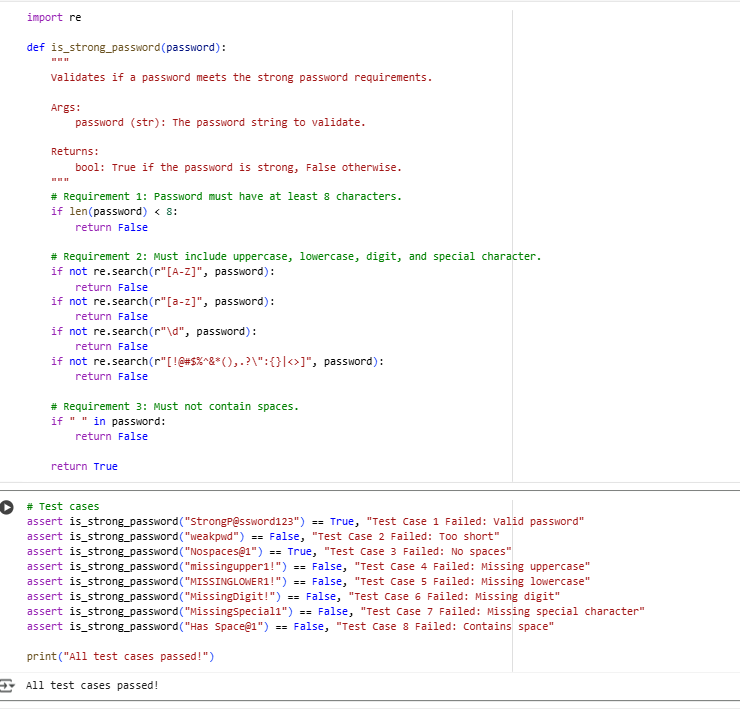
**NAME: P PRANAVI**

**ROLL:2403A51328**

**Batch:13**

**Task-1:**

* generate at least 3 assert test cases for is\_strong\_password(password) and implement the validator function



**Explanation:**

The function uses regular expressions (re module) to efficiently check for the presence of uppercase letters, lowercase letters, digits, and special characters. It also checks the length and for spaces. If all requirements are met, the function returns True; otherwise, it returns False.

**Task-2:**

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

****

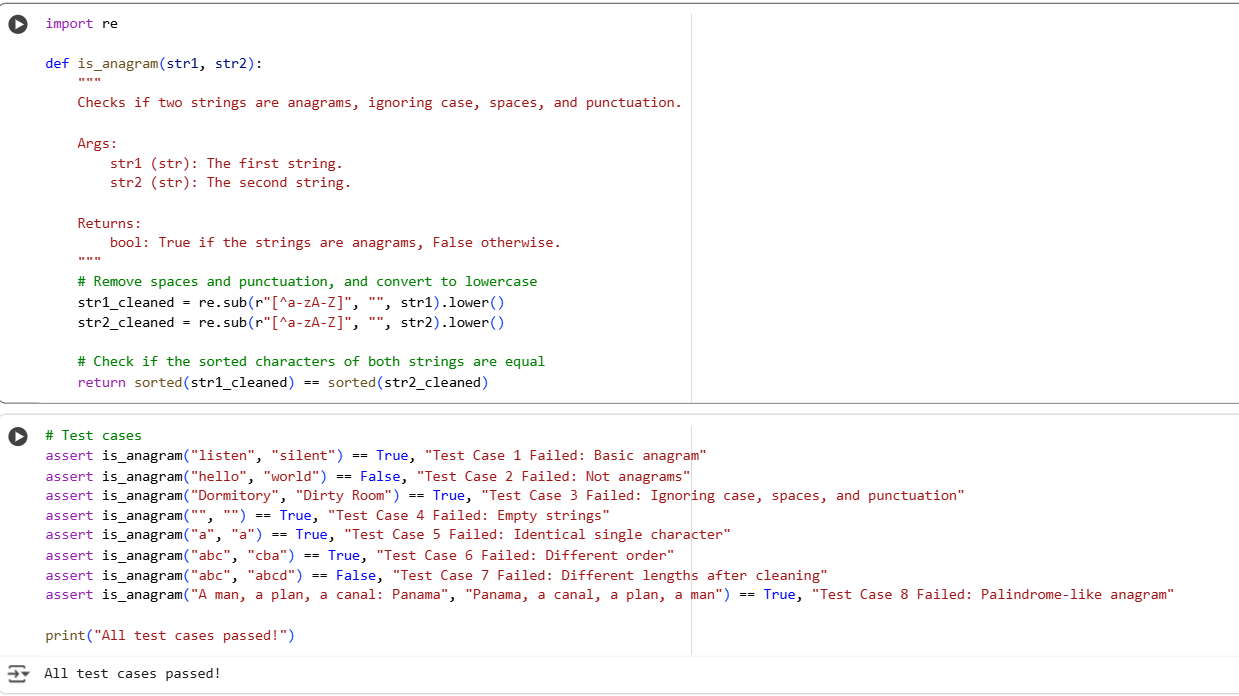
**Explanation:**

The implementation uses isinstance() to check if the input is an integer or a float. If it is, it then uses simple comparison operators (>, <, ==) to determine the classification. If the input is not an integer or float, it directly returns "Invalid Input".

The assert test cases are designed to confirm that the function correctly classifies various numbers (positive, negative, zero, integers, floats) and handles non-numeric inputs as specified. They also specifically test the boundary conditions of -1, 0, and 1.

**Task-3:**

generate at least 3 assert test cases for is\_anagram(str1, str2) and implement the function

****

**Explanation:**

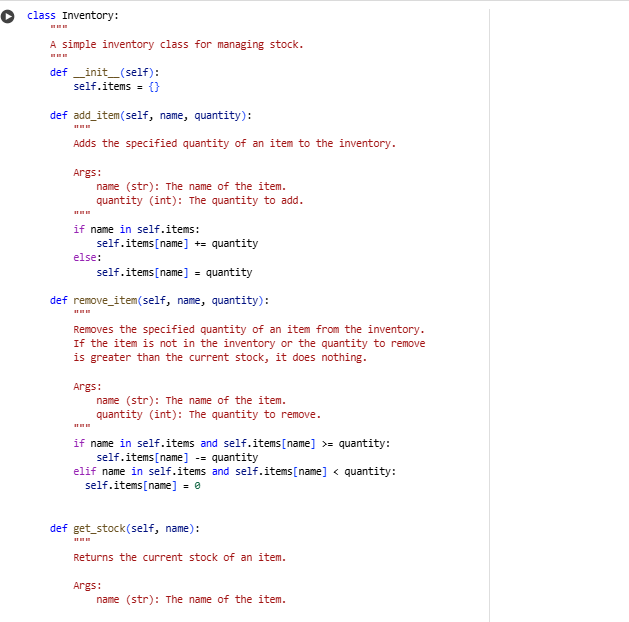
The "anagram" task involves creating a Python function named is\_anagram that takes two strings, str1 and str2, as input and determines if they are anagrams of each other.

The core idea of an anagram is that two words or phrases are anagrams if they contain the exact same characters, just in a different order.

The assert test cases are used to verify that the function works correctly with basic anagrams, non-anagrams, strings with different casing and spacing/punctuation, and the specified edge cases of empty and identical strings.

**Task-4:**

generate at least 3 assert-based tests for an Inventory class with stock management





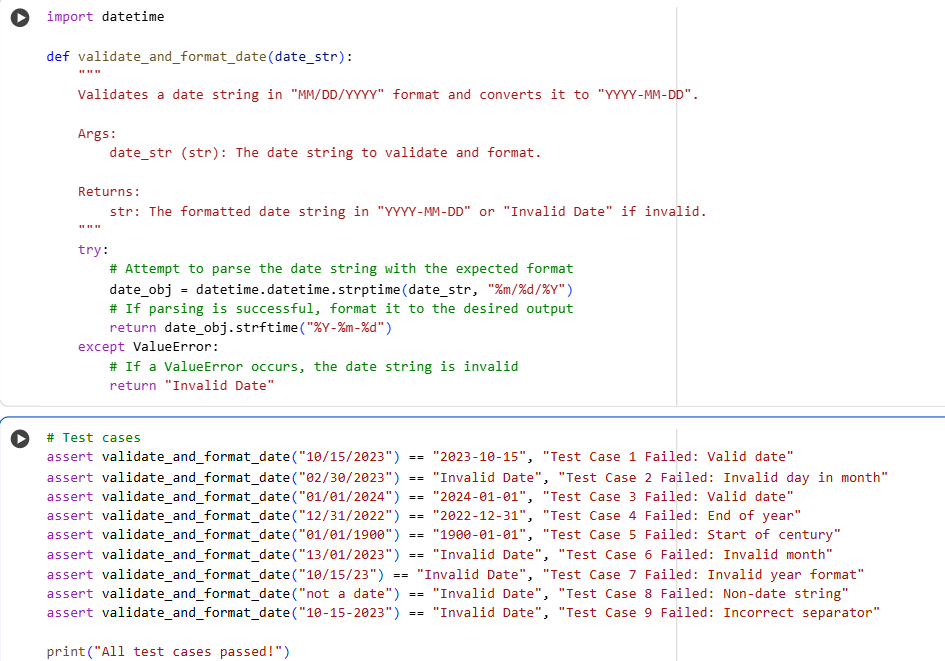
**Explanation:**

Inventory Class:

defines an Inventory class for managing stock. The \_init\_ method initializes an empty dictionary self.stock to hold item quantities. add\_item increases the quantity of an item; it handles adding new items or increasing existing stock. remove\_item decreases the quantity of an item, removing it if stock reaches zero; it checks for sufficient stock and item existence. get\_stock returns the current quantity of an item, or 0 if not found. Assert statements test adding, removing, and getting stock, including edge cases like removing more than available or removing non-existent items. "All assert tests passed!" is printed if all tests succeed.

**Task-5:**

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



Explanation:

defines validate\_and\_format\_date to process date strings. It imports the datetime class for date manipulation. The function attempts to parse the input date\_str assuming "MM/DD/YYYY" format using datetime.strptime. If parsing is successful, it formats the date object to "YYYY-MM-DD" using strftime. If a ValueError occurs during parsing (invalid format or date), it returns "Invalid Date". Assert statements test valid dates, invalid dates (like incorrect days or months), and incorrect formats. "All assert tests passed!" is printed if all tests pass.