### ****Task 1 – Password Strength Validator: AI Prompt****

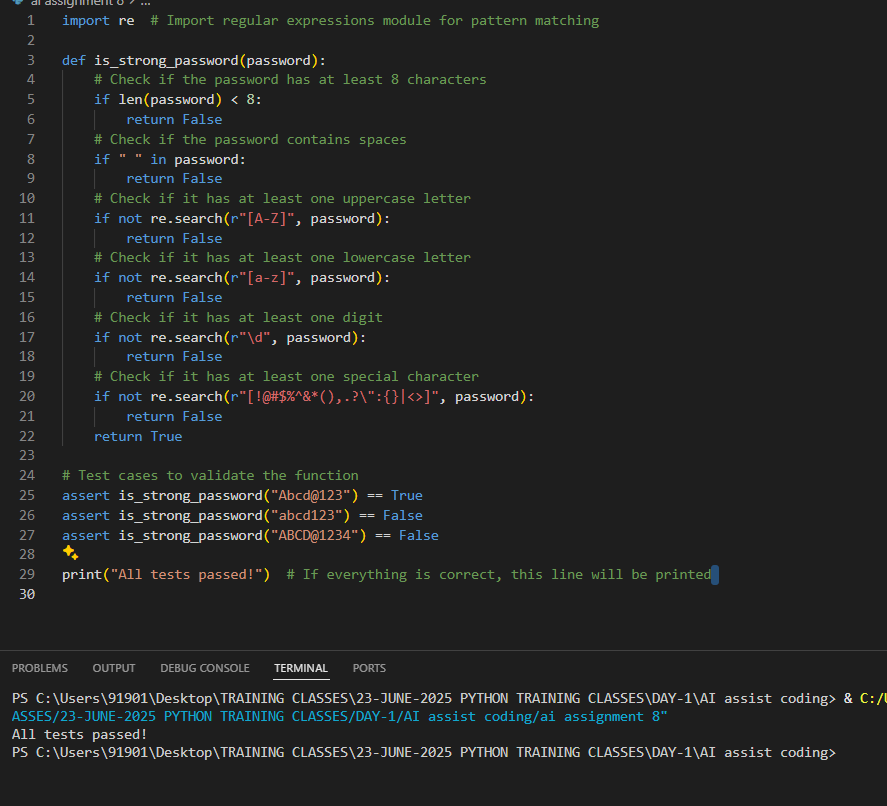
**Prompt:**  
“Generate at least 3 assert test cases for a Python function is\_strong\_password(password) that validates if a password is strong. The password rules are:

It must have at least 8 characters.

It must include at least one uppercase letter, one lowercase letter, one digit, and one special character (e.g., !@#$%^&\*(),.?":{}|<>).

It must not contain any spaces.  
Write test cases that check valid and invalid passwords, including edge cases.”

CODE AND OUTPUT:



## ****Task 2 – Number Classification with Loops****

### ****Question Description:****

Write a Python function called classify\_number(n) that classifies a number into one of the following categories:

**Positive** – if the number is greater than 0.

**Negative** – if the number is less than 0.

**Zero** – if the number is exactly 0.

Additionally, the function should handle invalid inputs gracefully by checking if the input is not a number (for example, if it’s a string or None) and returning "Invalid Input".

You must write at least 3 test cases using assert statements to cover:

Positive numbers

Negative numbers

Zero

Invalid inputs such as strings or None

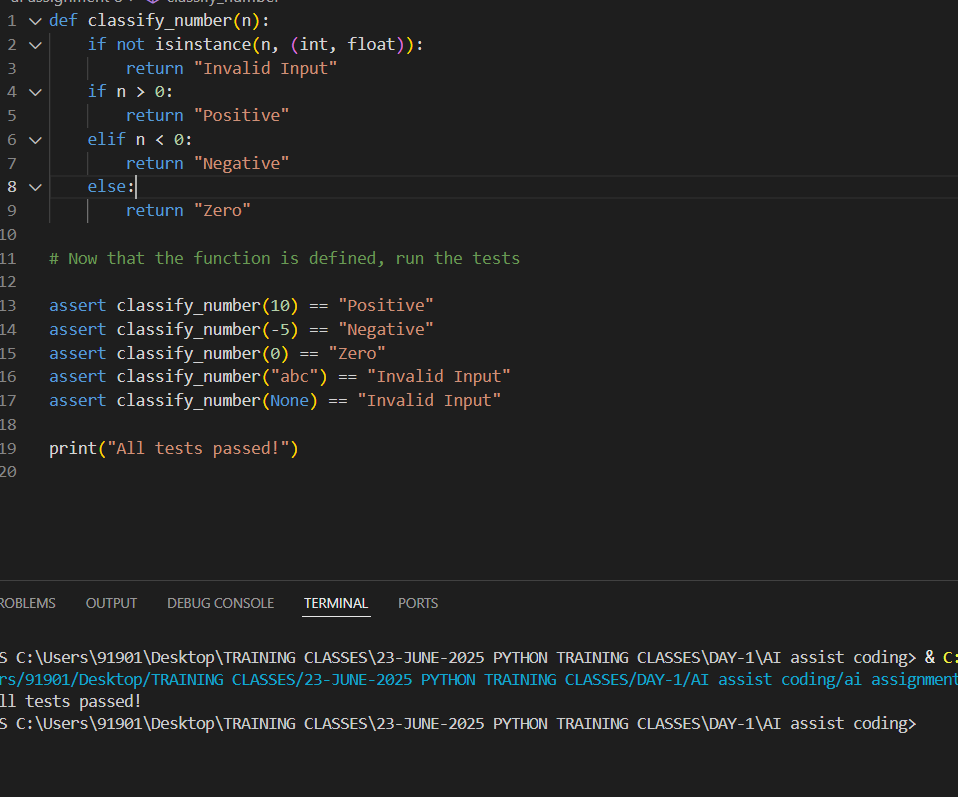
Boundary values like -1, 0, and 1

You should follow the Test-Driven Development (TDD) approach by writing test cases before implementing the function.

## ✅ ****AI Prompt for Task 2****

**Prompt:**  
“Generate at least 3 assert test cases for a Python function classify\_number(n) that classifies numbers as 'Positive', 'Negative', or 'Zero'. The function should also handle invalid inputs like strings and None, and test boundary cases such as -1, 0, and 1. Include valid and invalid inputs in the test cases.”

Code and output:



## ****Task 3 – Anagram Checker: Overview****

### ****Goal:****

Write a Python function is\_anagram(str1, str2) that checks if two strings are anagrams of each other.

Anagrams are words or phrases formed by rearranging the letters of another, ignoring:

Case differences (uppercase vs lowercase)

Spaces

Punctuation marks

You should also handle edge cases such as:

Empty strings

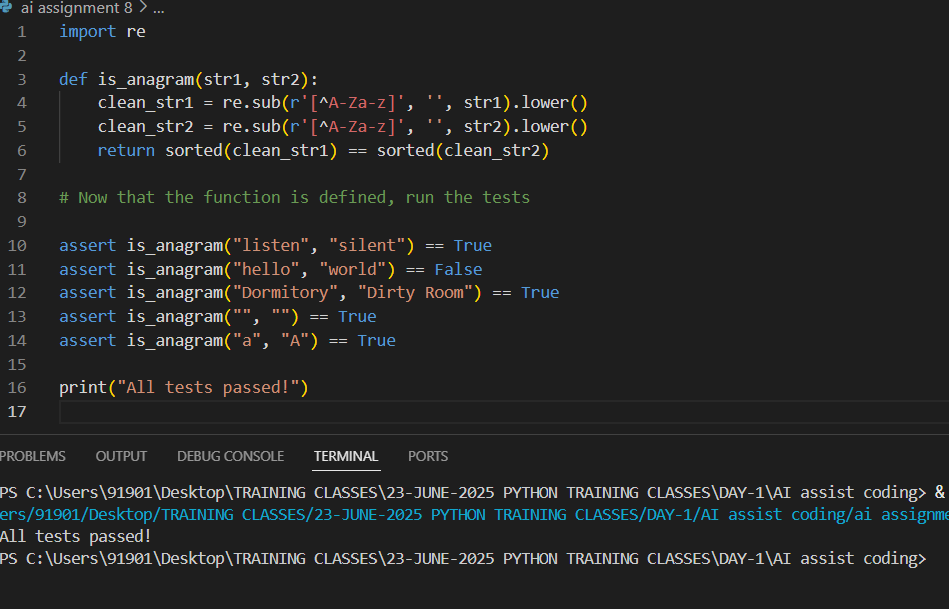
Identical words

You must write at least 3 test cases using assert statements to verify your function’s correctness.

## ✅ ****AI Prompt for Task 3****

**Prompt:**  
“Generate at least 3 assert test cases for a Python function is\_anagram(str1, str2) that checks whether two strings are anagrams of each other. The function should ignore case differences, spaces, and punctuation. Include edge cases such as empty strings and identical words in the test cases.”

Code & output:



## ****Task 4 – Inventory Class: Overview****

### ****Goal:****

Create a Python class called Inventory that simulates a real-world inventory system. The class should manage stock using the following methods:

add\_item(name, quantity) – adds a specified quantity of an item.

remove\_item(name, quantity) – removes a specified quantity of an item.

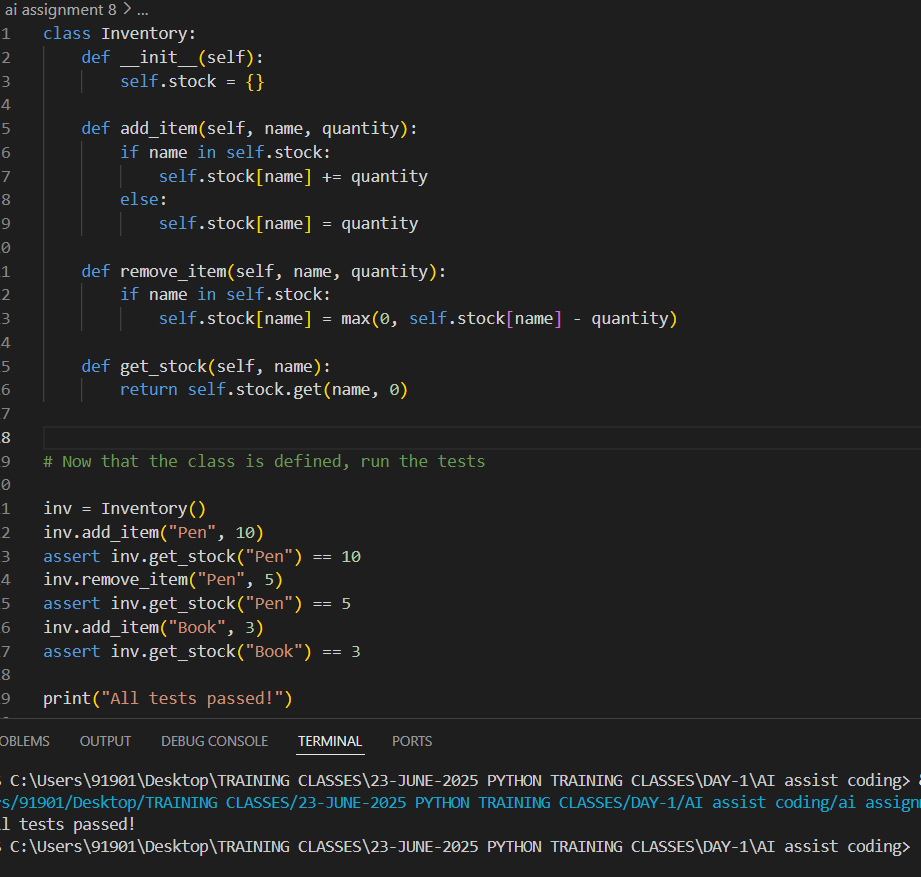
get\_stock(name) – returns the current stock of an item.

You need to write at least 3 test cases using assert to check that these methods work correctly.

## ✅ ****AI Prompt for Task 4****

**Prompt:**  
“Generate at least 3 assert test cases for a Python class Inventory with methods add\_item(name, quantity), remove\_item(name, quantity), and get\_stock(name). The test cases should check that items are added, removed, and their stock is updated correctly. Include cases where multiple items are managed.”

Code and output:



## ****Task 5 – Date Validation & Formatting: Overview****

### ****Goal:****

Write a Python function validate\_and\_format\_date(date\_str) that:

Validates if the input date is in the "MM/DD/YYYY" format.

Checks if the date is a valid calendar date.

Converts valid dates into the format "YYYY-MM-DD".

Returns "Invalid Date" for incorrect or invalid inputs.

You need to write at least 3 test cases using assert to cover:

Correctly formatted dates

Invalid dates (like February 30)

Edge cases like leap years, invalid formats, etc.

## ✅ ****AI Prompt for Task 5****

**Prompt:**  
“Generate at least 3 assert test cases for a Python function validate\_and\_format\_date(date\_str) that validates dates in the 'MM/DD/YYYY' format and converts valid dates to 'YYYY-MM-DD'. It should handle invalid dates and formats gracefully by returning 'Invalid Date'. Include edge cases like invalid days and correct leap year handling.”

Code and output:

