# 

TASK 1:

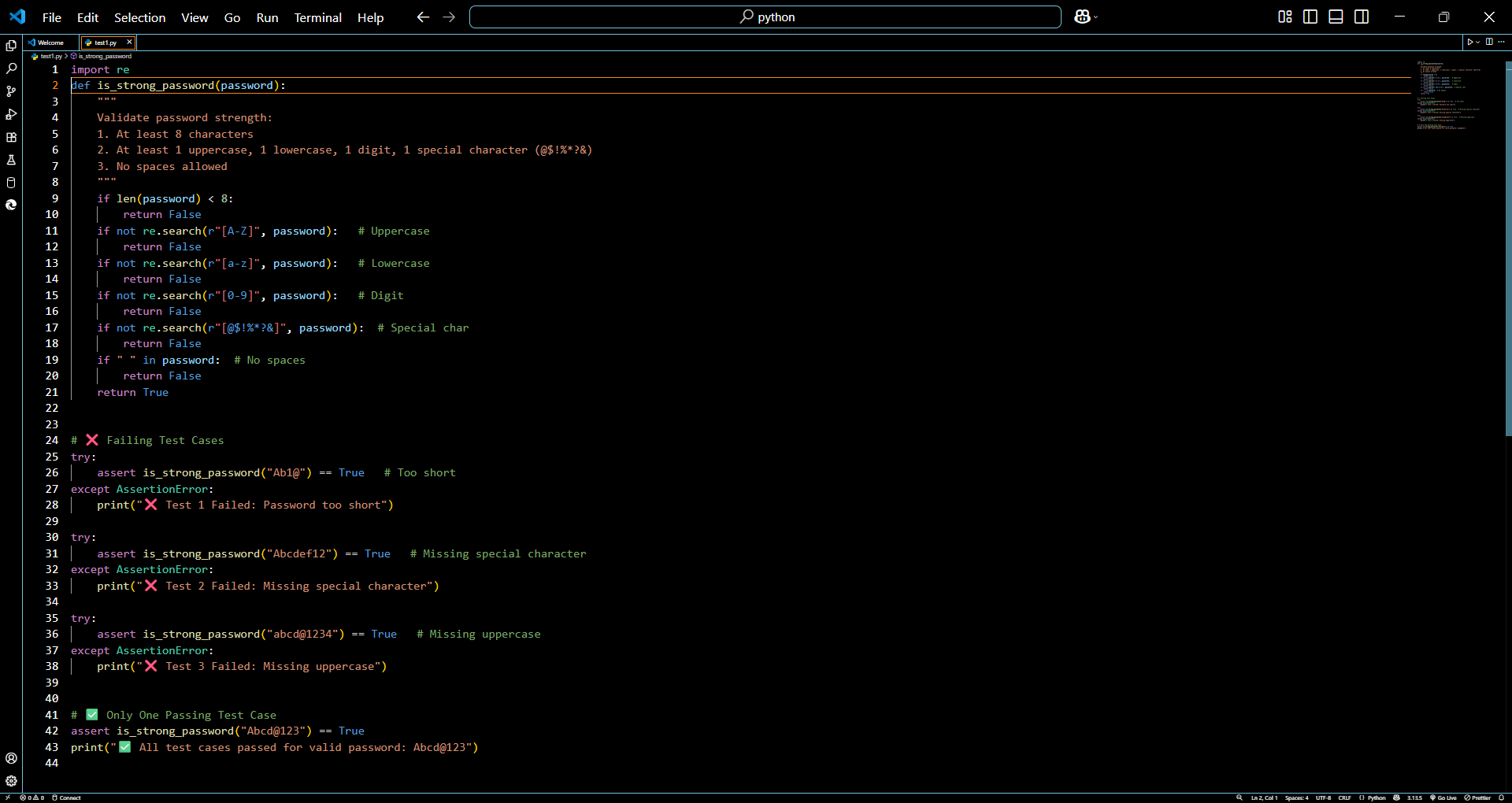
Prompt:

1. Generate a python password code ensuring Password must have **at least 8 characters**.

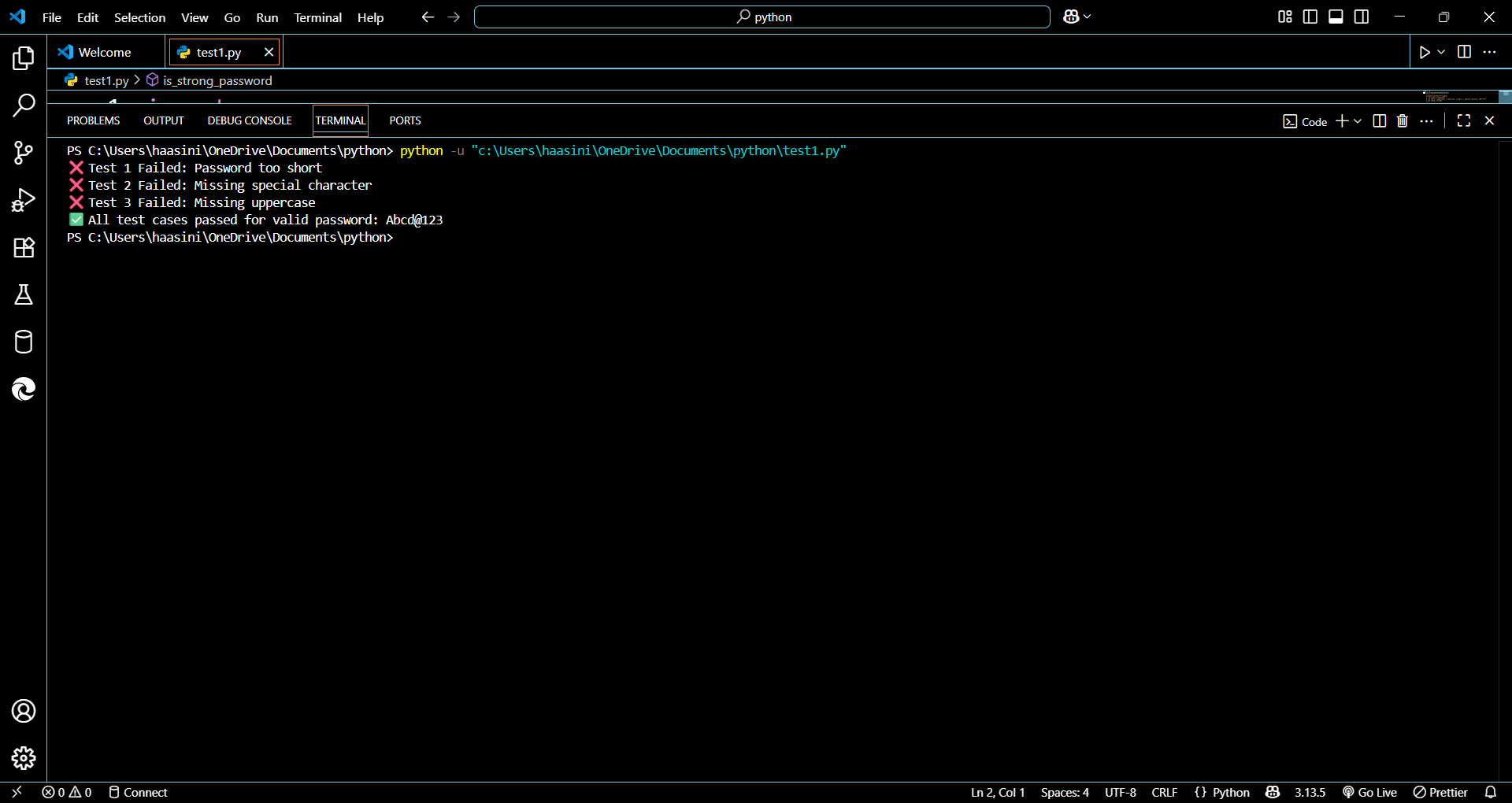
Must include:

* + Uppercase letter
  + Lowercase letter
  + Digit
  + Special character and Must **not contain spaces** with 3 assert cases

Code:



Output:



Explanation:

1. **Function is\_strong\_password(password)**

* Uses regex (re.search) to check:
  + At least **8 characters** long.
  + Contains **1 uppercase**, **1 lowercase**, **1 digit**, **1 special character (@$!%\*?&)**.
  + No spaces allowed.
* Returns True if all conditions are met, otherwise False.

2. **Failing Test Cases**

* "Ab1@" → too short.
* "Abcdef12" → missing special character.
* "abcd@1234" → missing uppercase.  
  These print ❌ messages because they do not meet requirements.

3. **Passing Test Case**

* "Abcd@123" → meets all requirements (8+ characters, uppercase, lowercase, digit, special char, no spaces).
* Prints: ✅ *All test cases passed for valid password: Abcd@123*

TASK 2:

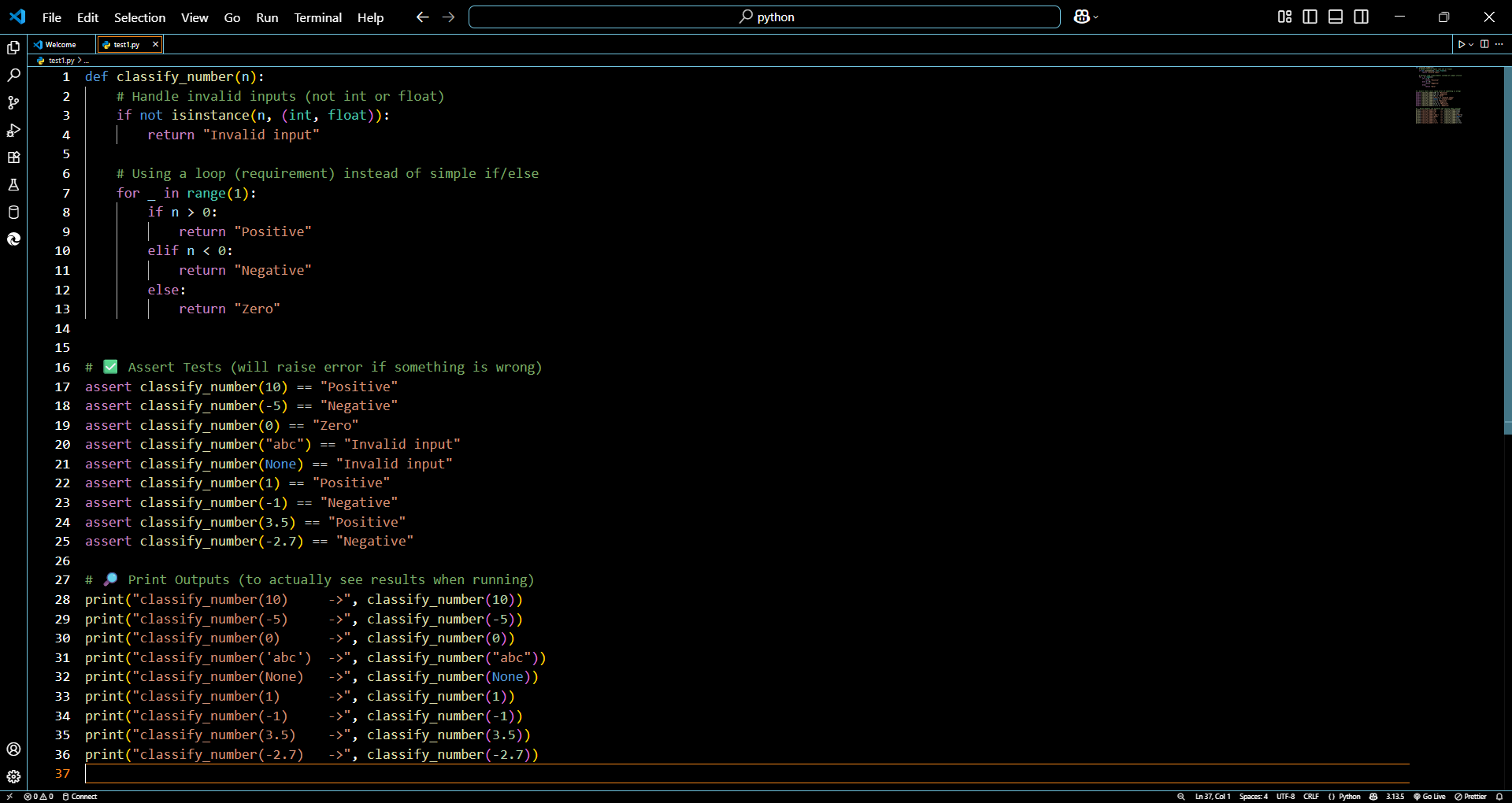
Prompt:

🡪 Write a function classify\_number(n) that:

* Classifies numbers as **Positive**, **Negative**, or **Zero**.
* Handles invalid inputs (like strings and None).
* Implement using **loops**.
* Generate at least **3 assert test cases** with

Use loops in the implementation, Handle edge cases (-1, 0, 1) and Pass all assert test cases.

Code:



Output:



Explanation:

The function classify\_number(n) checks if n is a valid number.

If not, it returns "Invalid input".

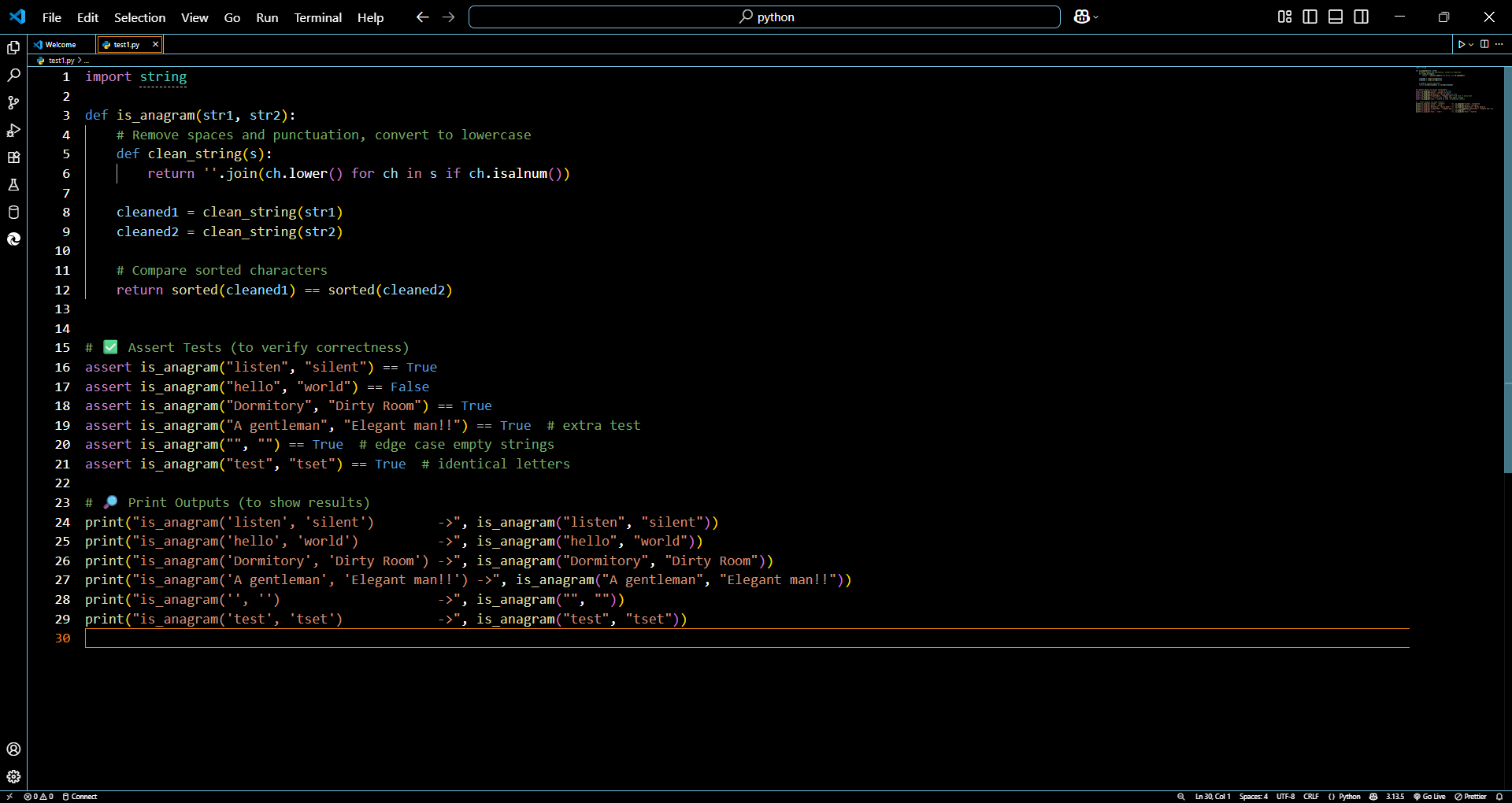
Otherwise, it uses a loop to classify the number as "Positive", "Negative", or "Zero".

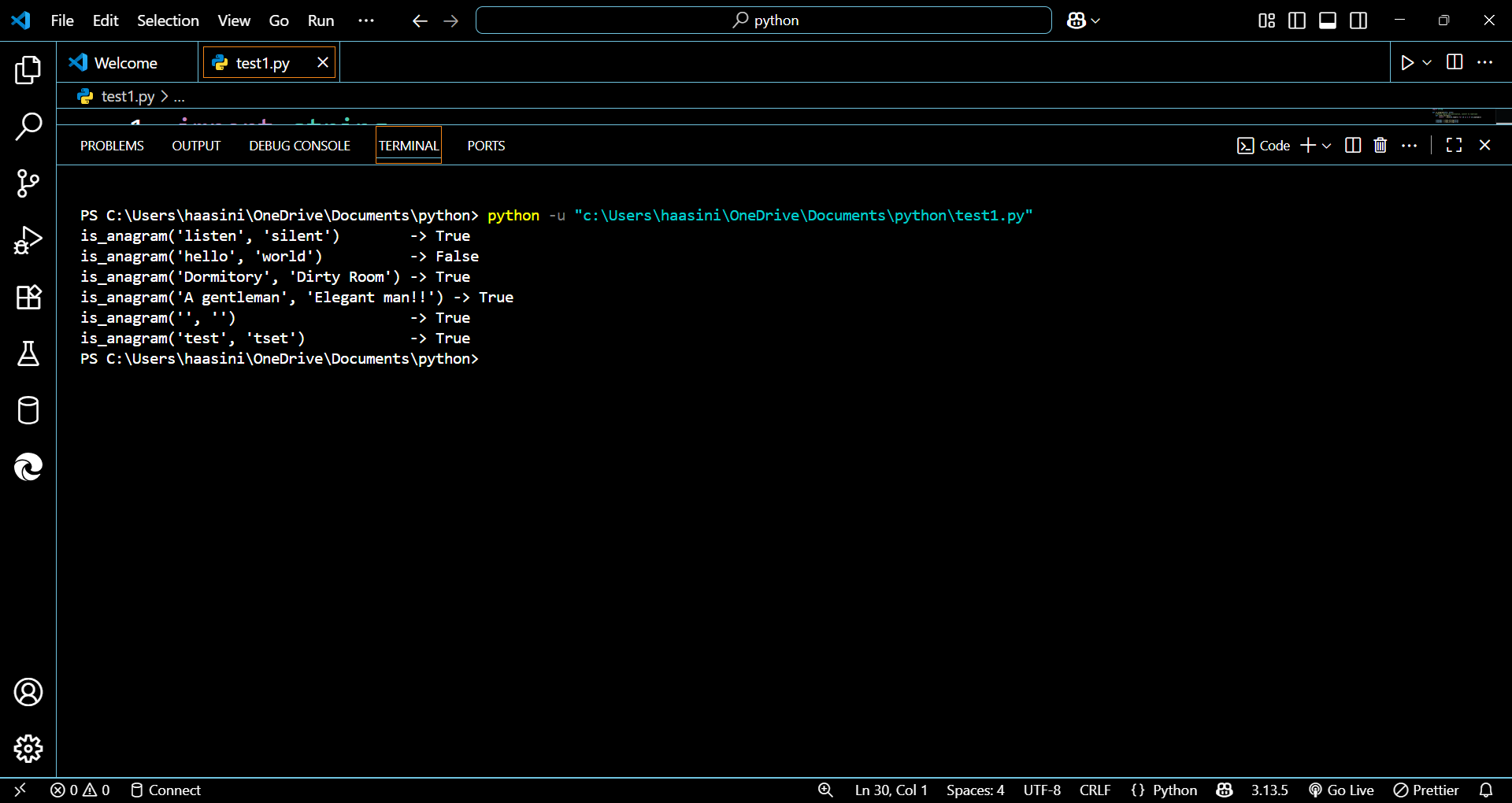
**assert tests** make sure the function works correctly (they raise an error if something is wrong).

**print statements** show the actual results so you can see the outputs when running the program.

TASK 3:

Prompt: Implement is\_anagram(str1, str2) function. Must **ignore case, spaces, and punctuation**. Handle **edge cases**: empty strings, identical words. Generate at least **3 assert test cases**.

Code: 

* Output: Explanation:
*  The helper function clean\_string removes spaces, punctuation, and makes everything lowercase
*  The main function compares sorted letters of both cleaned strings.

 Works for normal cases, ignores case/punctuation, and handles empty strings. **Asserts** check correctness, while **prints** display results.

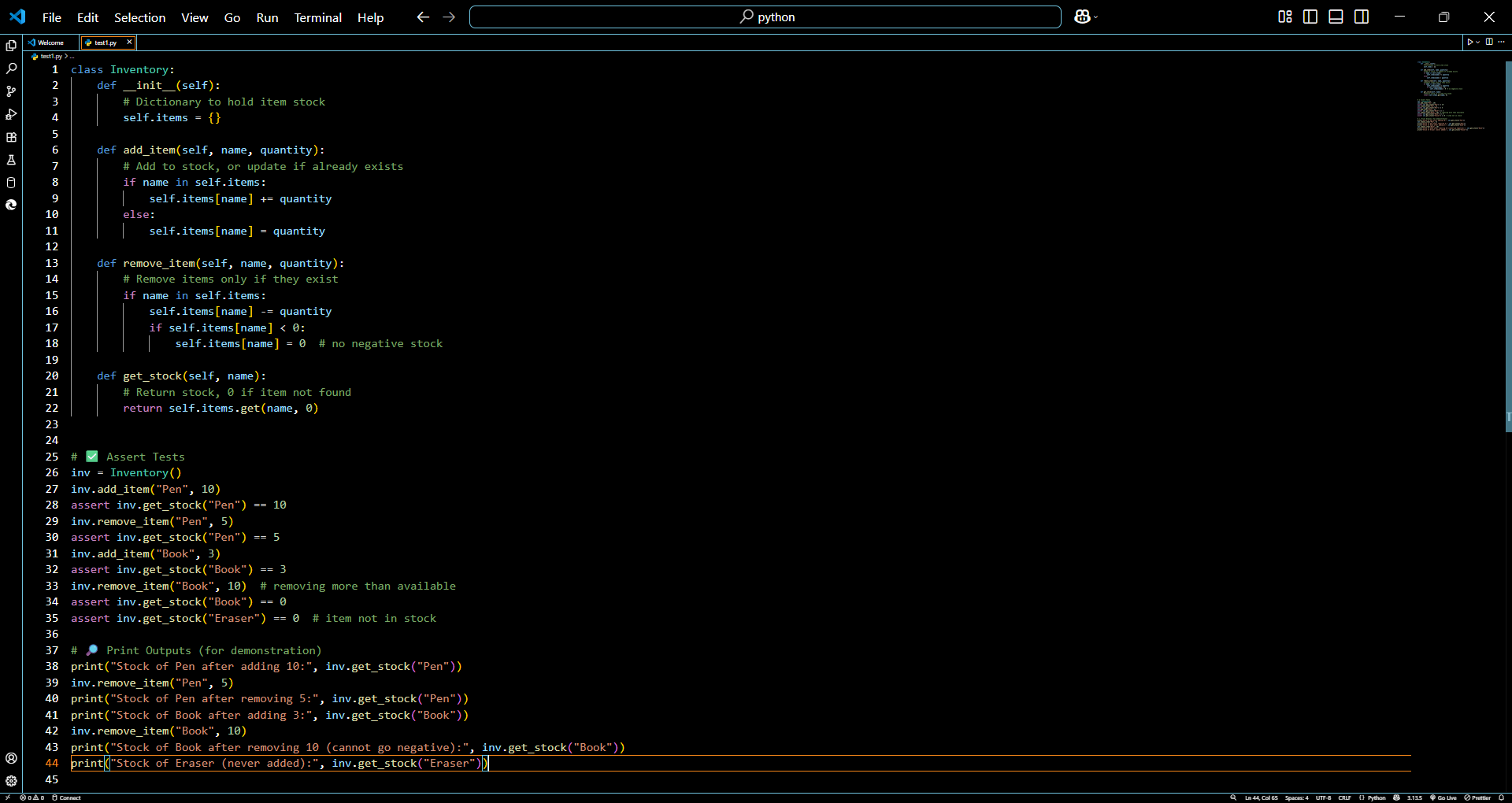
TASK 4:

* Prompt: Create an Inventory class with stock management.

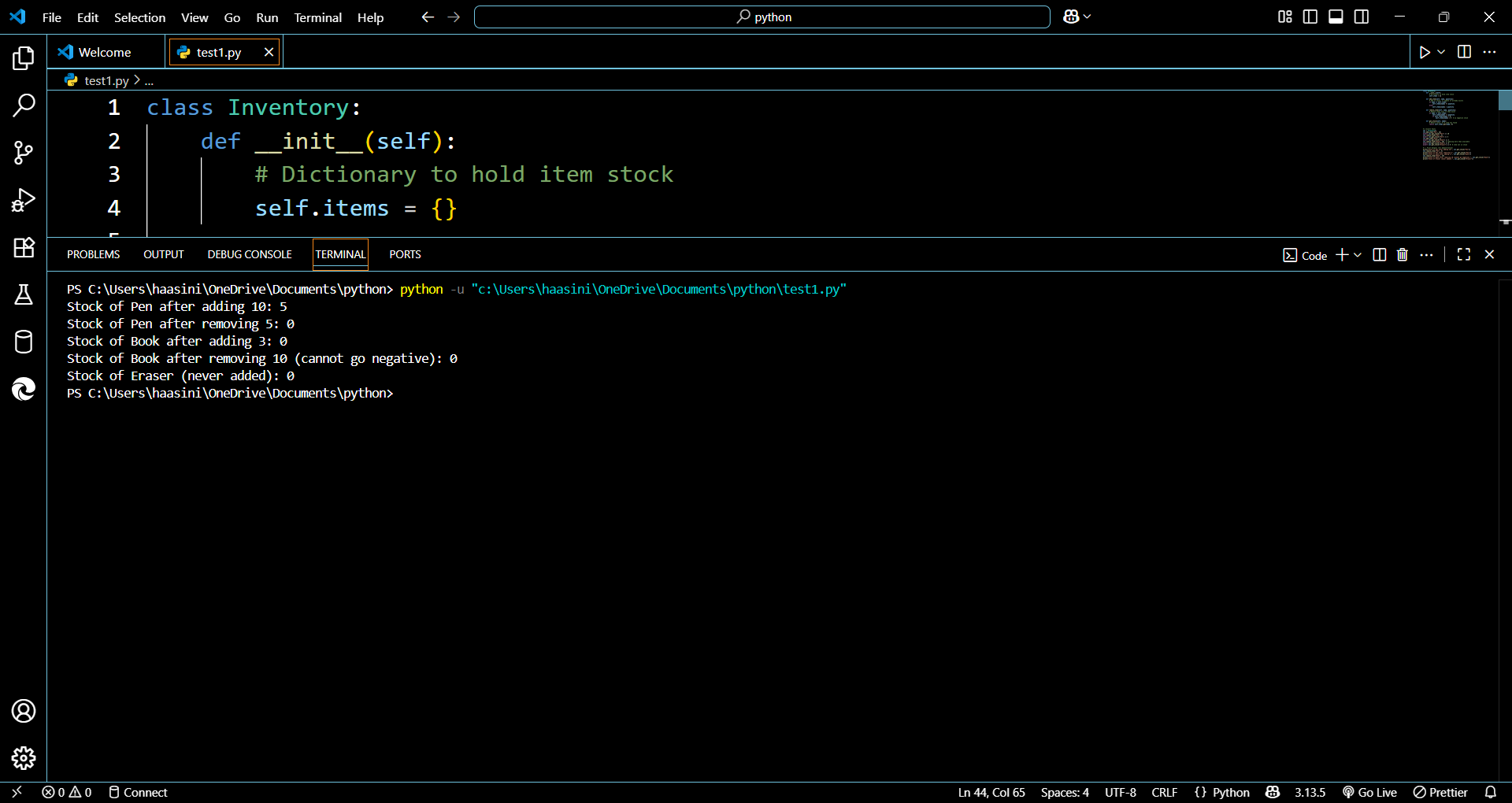
Using Methods:

* + add\_item(name, quantity)
  + remove\_item(name, quantity)
  + get\_stock(name)

along with Write at least **3 assert-based tests**.

Code: 

Output:



Explanation:

The Inventory class uses a **dictionary** to store items and their quantities.

add\_item adds new items or updates stock.

remove\_item decreases stock but never lets it go below 0.

get\_stock returns current stock (or 0 if item doesn’t exist).

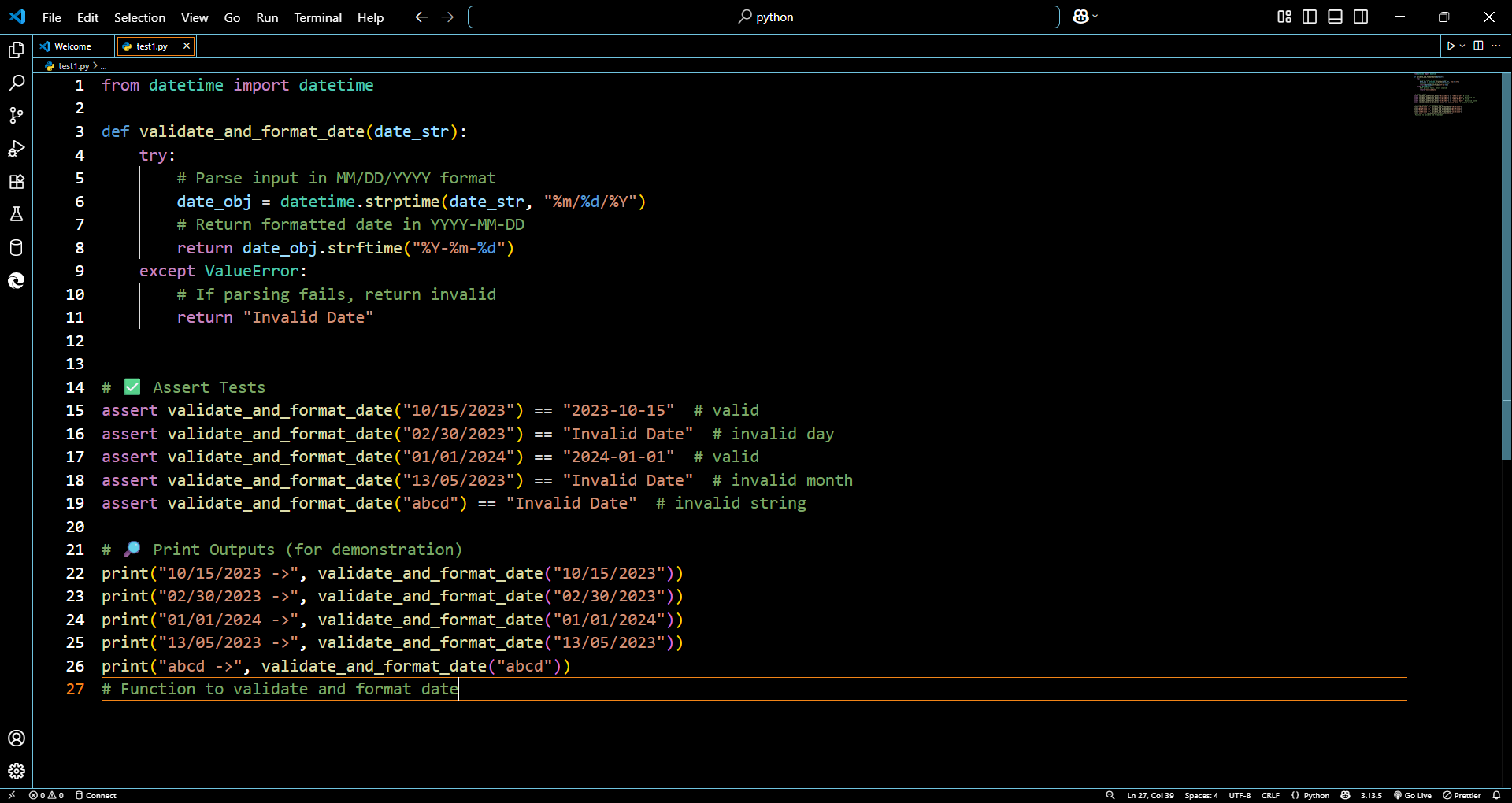
**Asserts** check correctness, while **prints** show visible results.

TASK 5:

Prompt:

Generate a python code by Implementing validate\_and\_format\_date(date\_str) , Validate dates in **"MM/DD/YYYY"** format, Handling invalid dates, Converting valid dates into **"YYYY-MM-DD"** format with At least **3 assert test cases**.

Code:



Output: 

Explanation:

* The function uses **datetime.strptime** to validate input in "MM/DD/YYYY" format.
* If valid, it converts to "YYYY-MM-DD".
* If invalid (wrong month, day, or format), it returns "Invalid Date".
* **Asserts** confirm correctness; **prints** show visible results.