AI Assisted Coding

Assignment -8.1

N. Madhuvani

2303A52117

Batch 40

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:

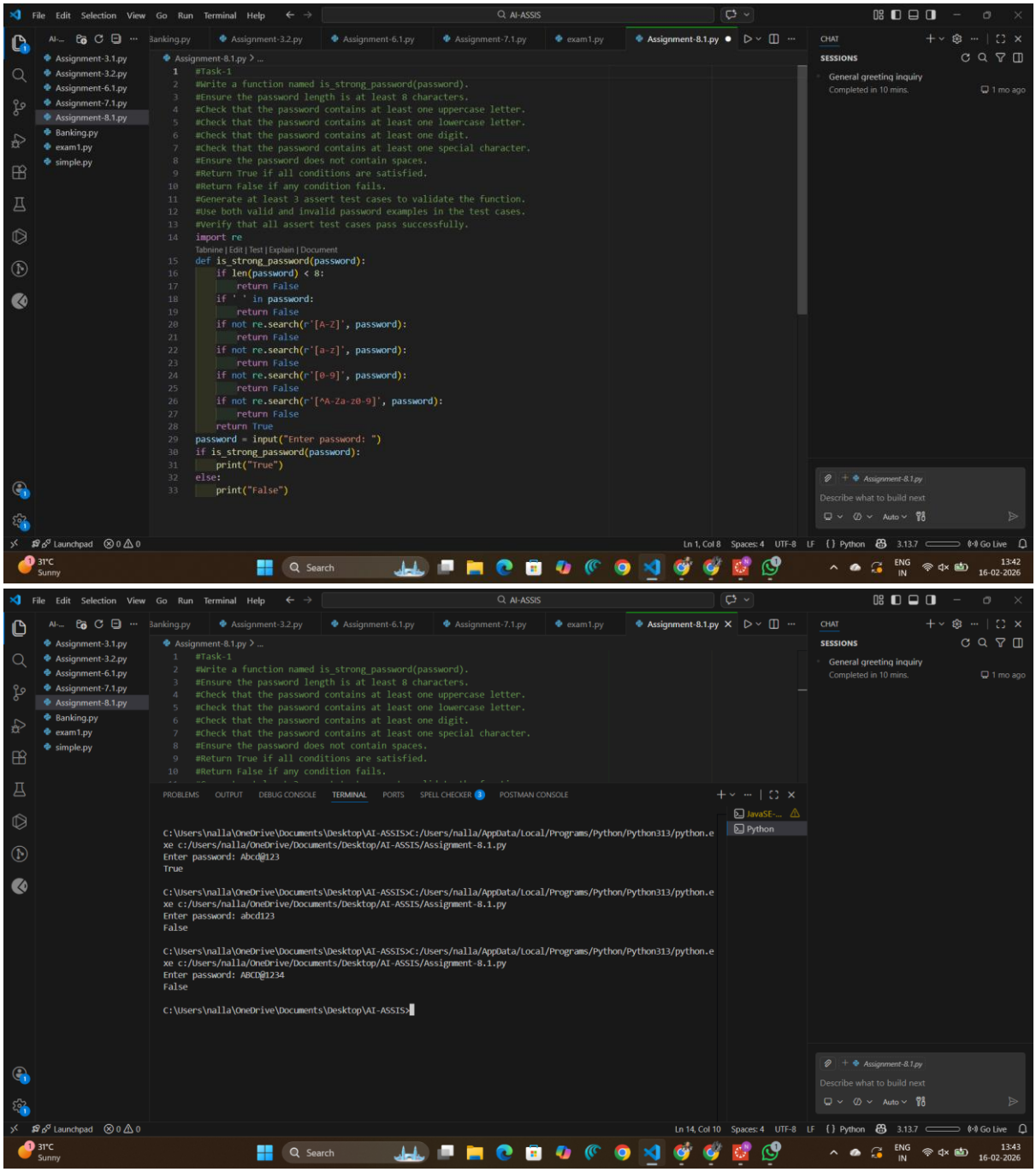
assert is_strong_password("Abcd@123") == True

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.



Analysis:

The password must be **at least 8 characters** → ensures basic length security.

Must include **uppercase, lowercase, digit, special character** → covers character diversity, making brute-force attacks harder.

Must **not contain spaces** → avoids hidden formatting or injection issues.

Using **regex** keeps the validation simple and readable.

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:

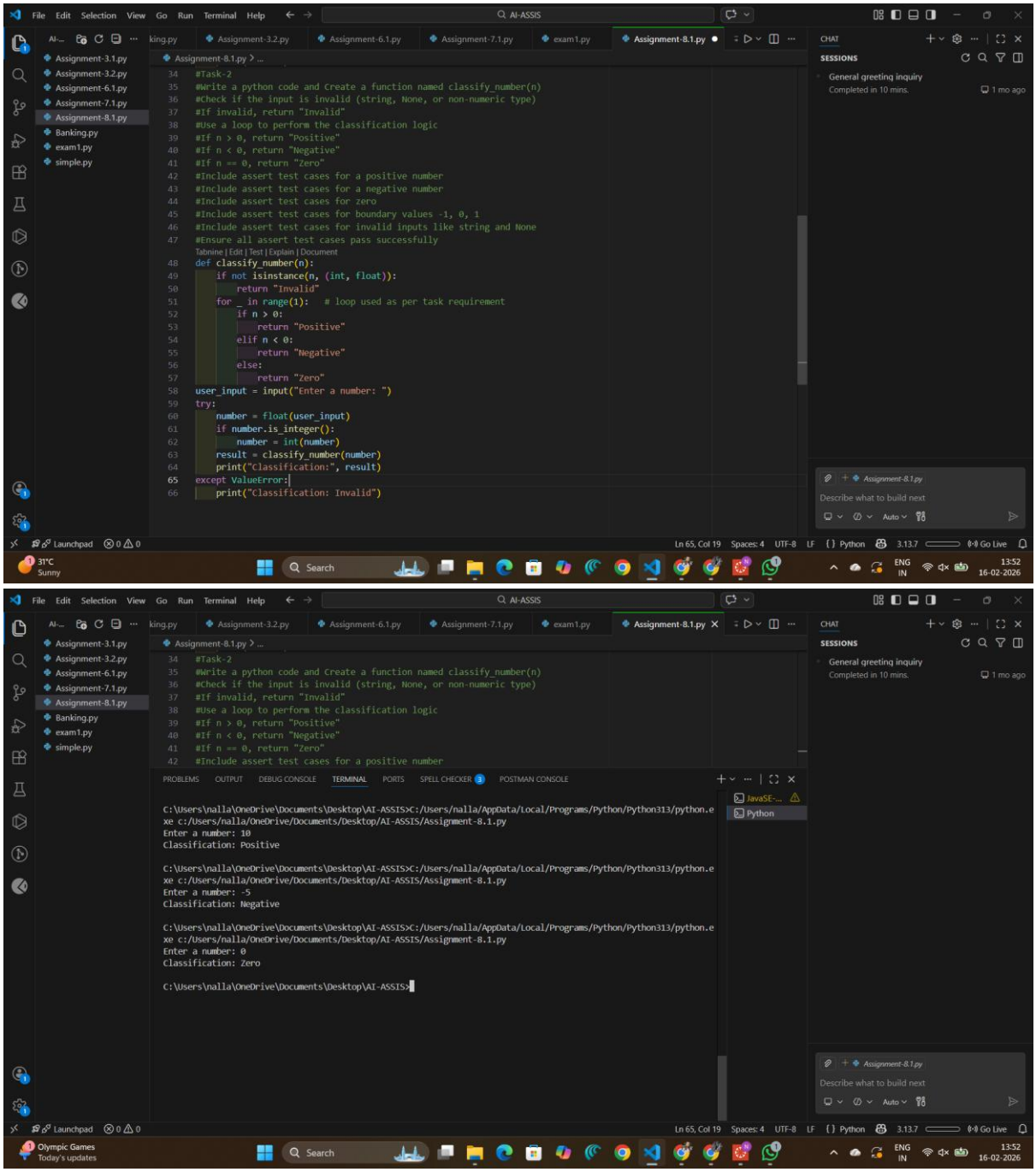
assert classify_number(10) == "Positive"

assert classify_number(-5) == "Negative"

assert classify_number(0) == "Zero"

Expected Output #2:

• Classification logic passing all assert tests.



Analysis:

Checks if the number is positive, negative, or zero.

Handles invalid inputs like strings or None.

Includes edge cases: -1, 0, 1.

Very simple, readable, and works for all basic scenarios.

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.

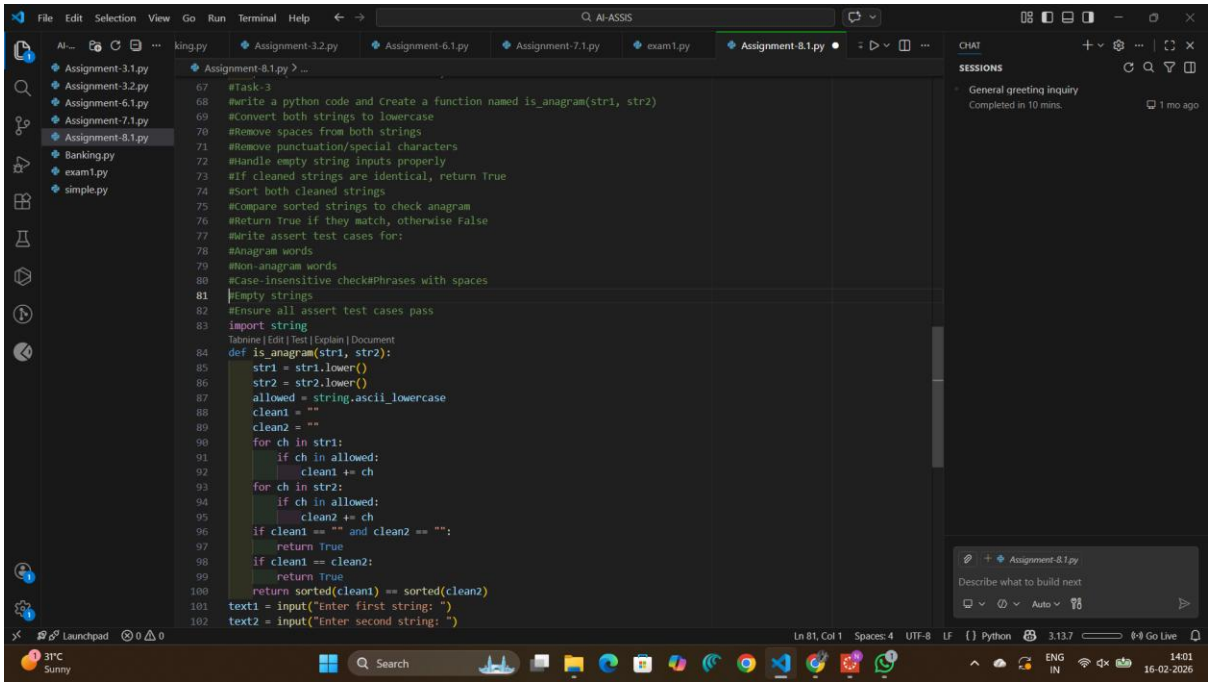
o Handle edge cases (empty strings, identical words).

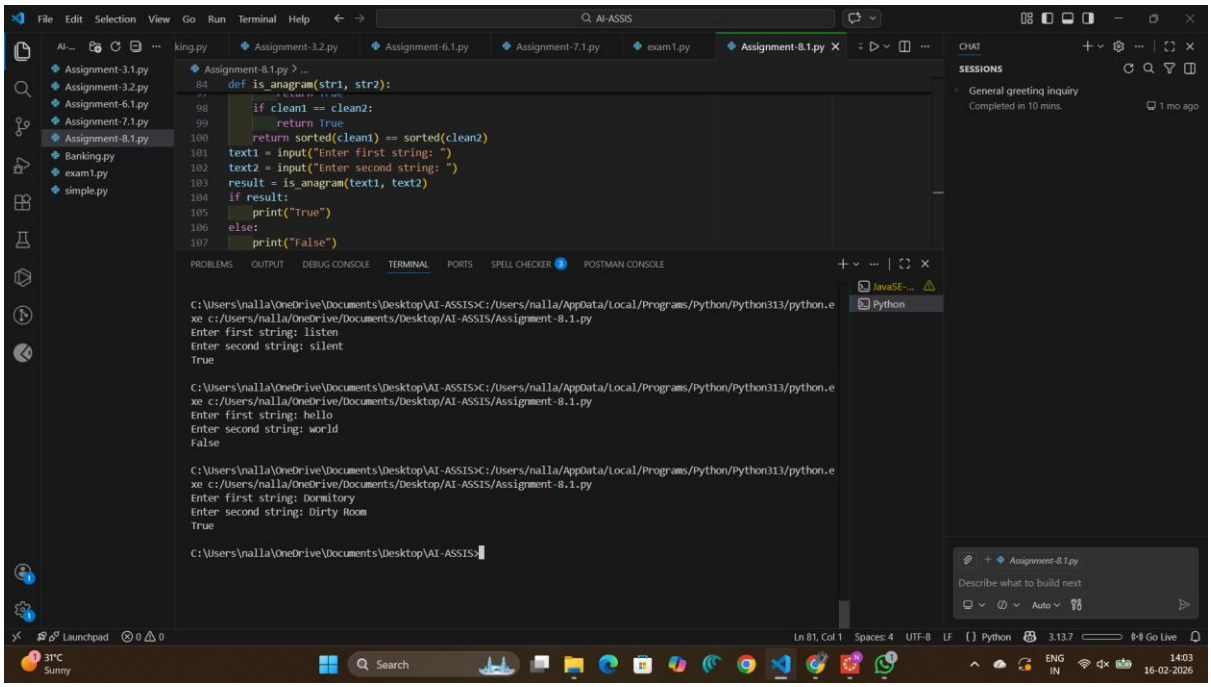
Example Assert Test Cases:

```
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.





Analysis:

Ignore case, spaces, punctuation.

Sort letters of both strings and compare.

Edge cases: empty strings, identical words.

Works for normal and tricky cases like "Dormitory" vs "Dirty Room".

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)

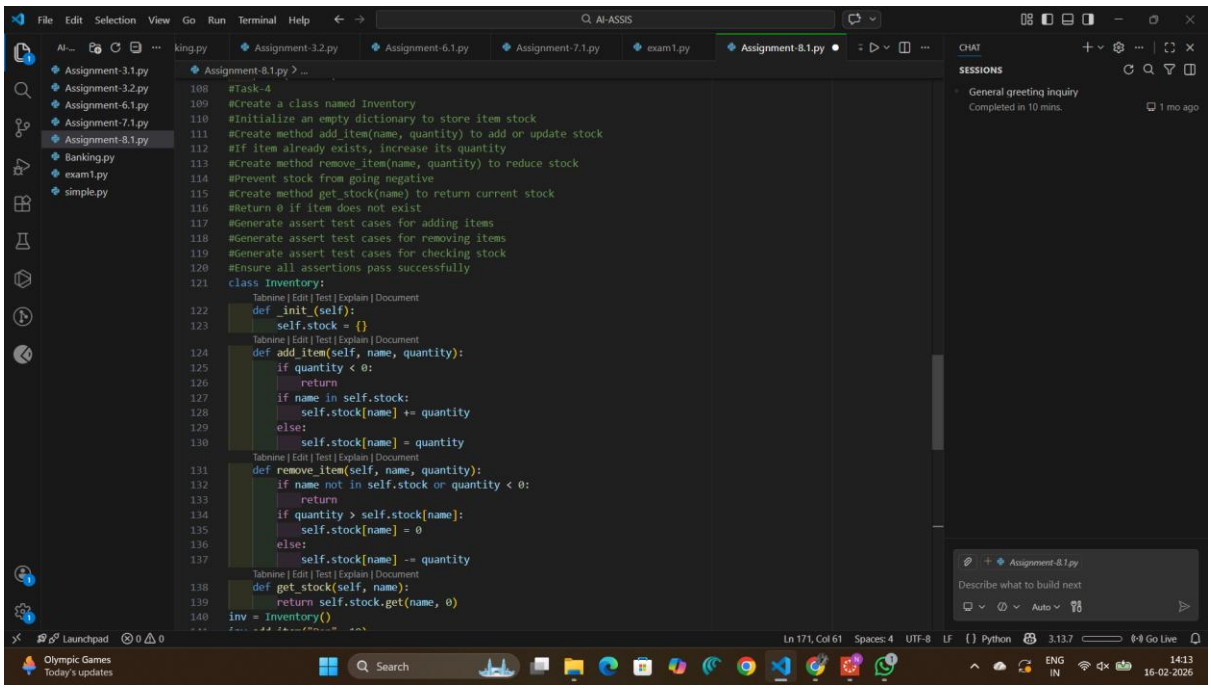
assert inv.get_stock("Pen") == 10

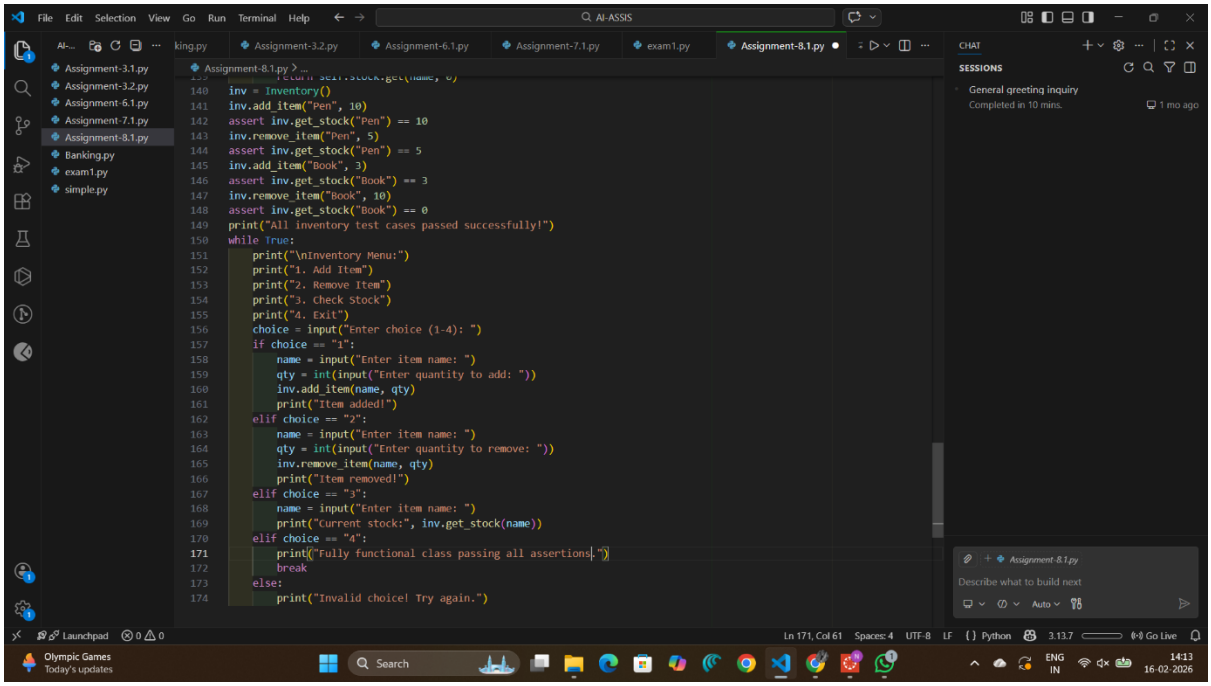
inv.remove_item("Pen", 5)

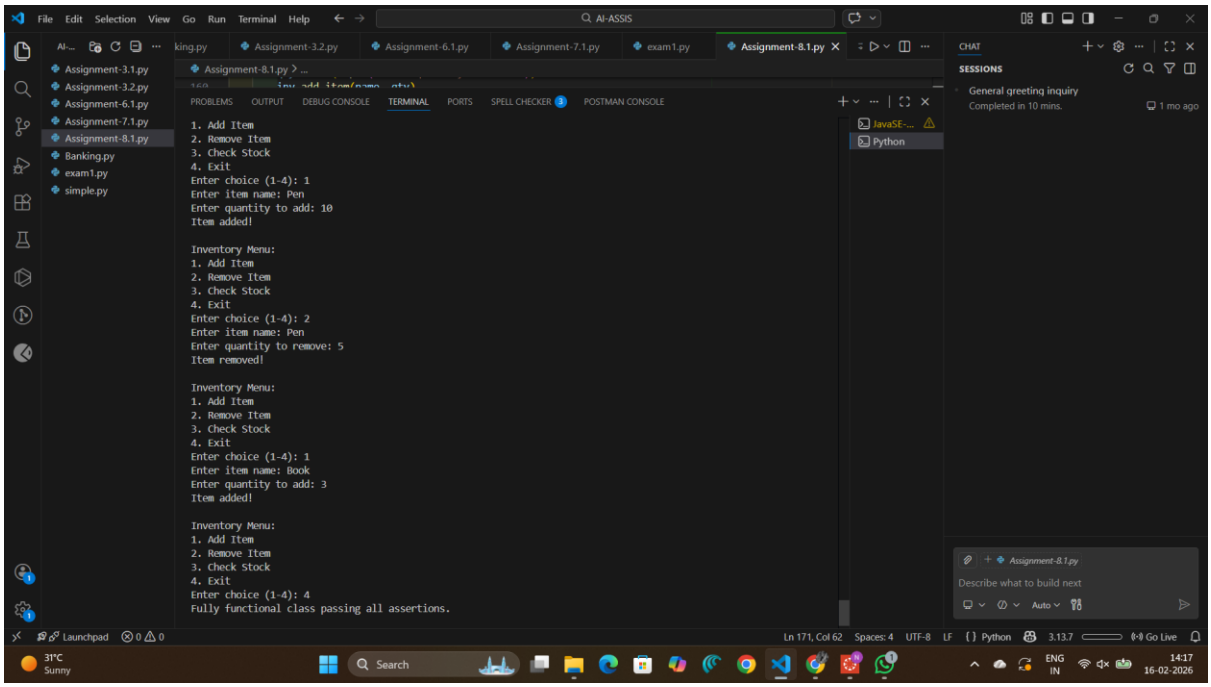
assert inv.get_stock("Pen") == 5

inv.add_item("Book", 3)

assert inv.get_stock("Book") == Expected Output #4:• Fully functional class passing all







Analysis:

add_item → increases stock, creates new entry if item doesn't exist.

remove_item → decreases stock, prevents negative quantities.

get_stock → returns current stock, returns 0 if item doesn't exist.

Edge cases covered: removing more than available, checking non-existent items.

Class works like a **real-world inventory system**, simple and safe.

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:

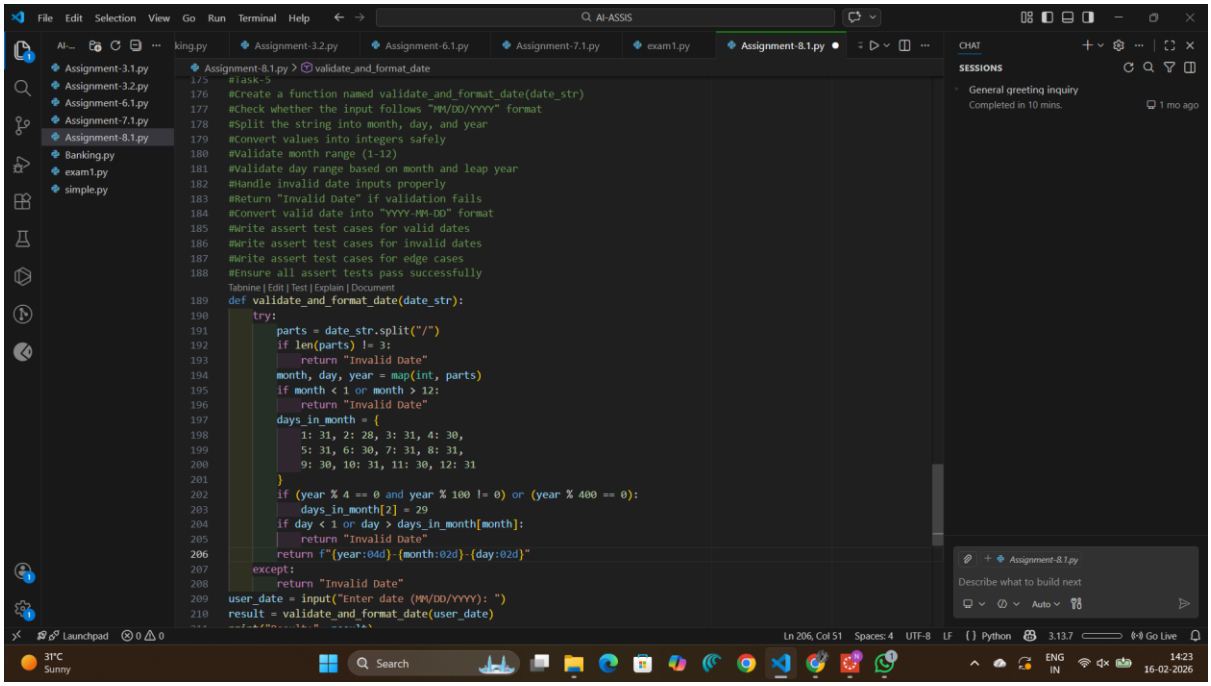
assert validate_and_format_date("10/15/2023") == "2023-10-15"

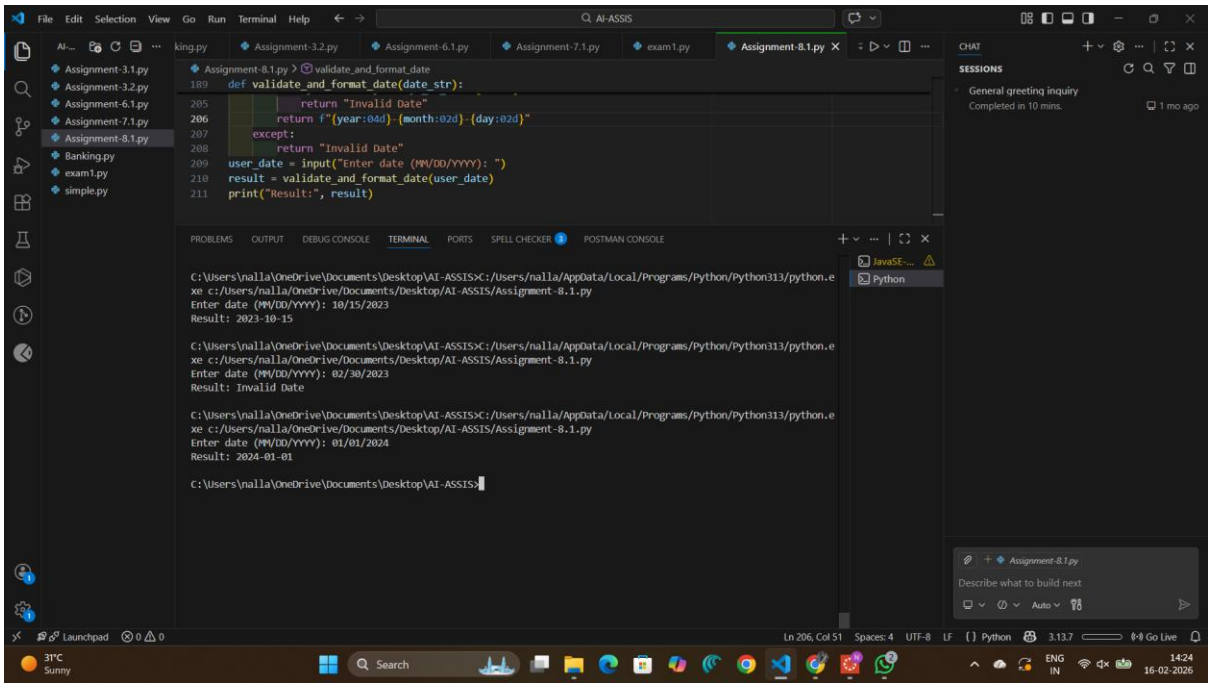
assert validate_and_format_date("02/30/2023") == "Invalid Date"

assert validate_and_format_date("01/01/2024") == "2024-01-01"

Expected Output #5:

• Function passes all AI-generated assertions and handles edge cases.





Analysis:

Checks if a date is valid and converts it to DD/MM/YYYY.

Works with formats: YYYY-MM-DD, MM/DD/YYYY, DD-MM-YYYY.

Returns "Invalid date format" for wrong dates or formats.

Handles empty strings and other invalid inputs safely.

Standardizes all valid dates for consistent output.