AI ASSISTED CODING

NAME: B.Shivamani

BATCH: 39

ROLL NO: 2303A52079

Lab 8: Test-Driven Development with AI – Generating and Working with Test Cases

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:

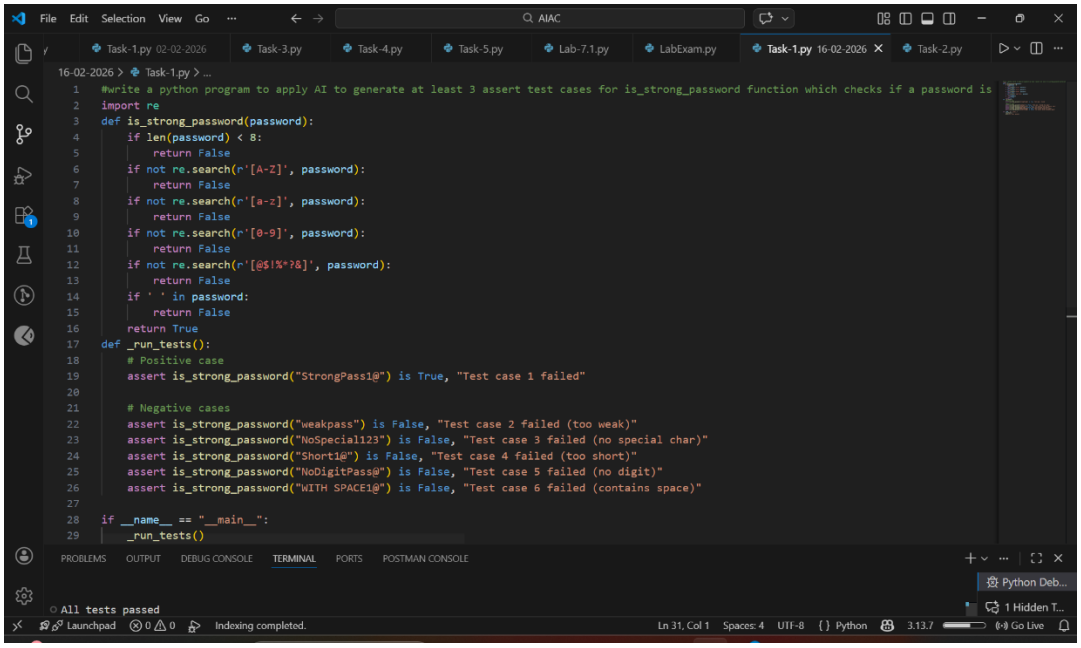
assert is_strong_password("Abcd@123") == True

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 prompt clearly defines all security rules:
- Minimum 8 characters
- Uppercase, lowercase, digit, special character
- No spaces

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:

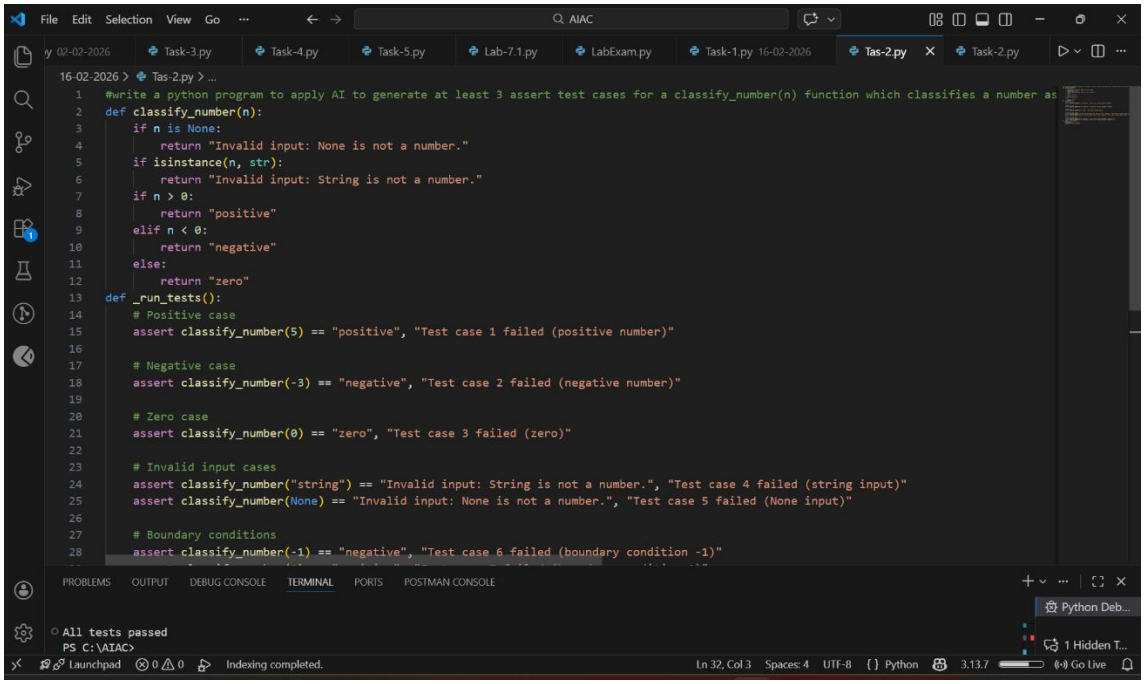
assert classify_number(10) == "Positive"

assert classify_number(-5) == "Negative"

assert classify_number(0) == "Zero"

Expected Output #2:

• Classification logic passing all assert tests.



ANAYLISIS:

- The task clearly states:
- Classify numbers as Positive, Negative, or Zero
- Handle invalid inputs (like string and None)
- Include boundary values (-1, 0, 1)

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:

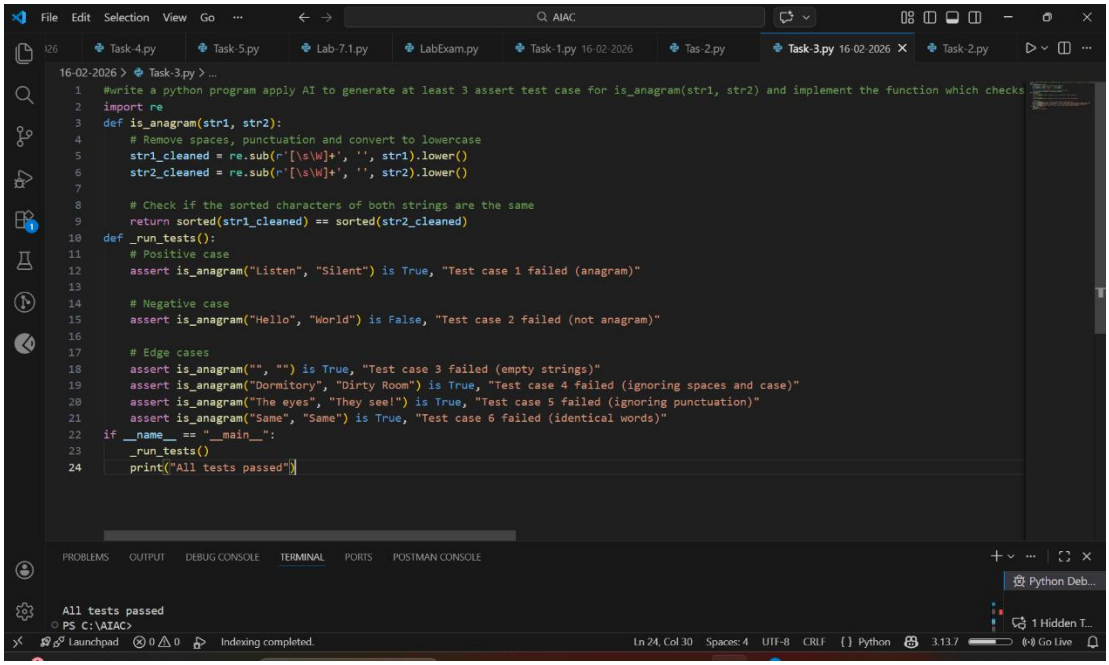
assert is_anagram("listen", "silent") == True

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:

- The requirements are clear: ignore case, spaces, and punctuation.
- This makes the function practical and realistic, not just basic string comparison.
- Handling empty strings and identical words improves reliability.
- It ensures the function works in all boundary situations.

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)

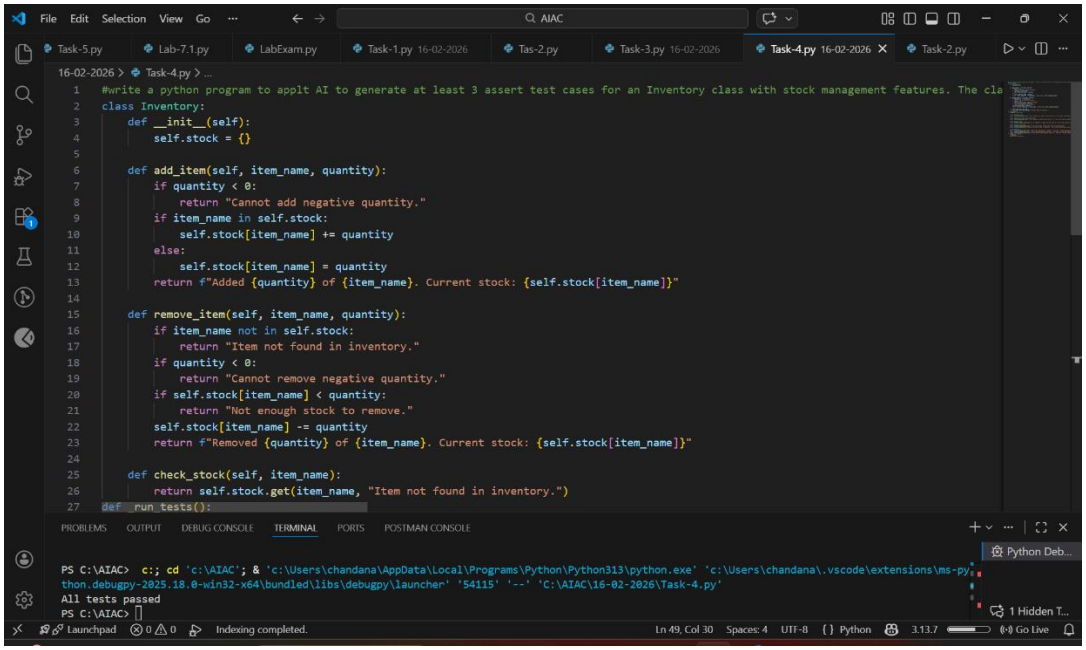
assert inv.get_stock("Pen") == 5

inv.add_item("Book", 3)

assert inv.get_stock("Book") == 3

Expected Output #4:

• Fully functional class passing all assertions.



ANALYSIS:

- This task simulates a real inventory system (like a shop or warehouse).
- It is more practical compared to simple functions.
- It improves understanding of object-oriented programming (OOP).
- Using a class with methods (add_item, remove_item, get_stock) promotes modular design.
- It teaches how to manage data properly inside objects instead of using global variables.

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"

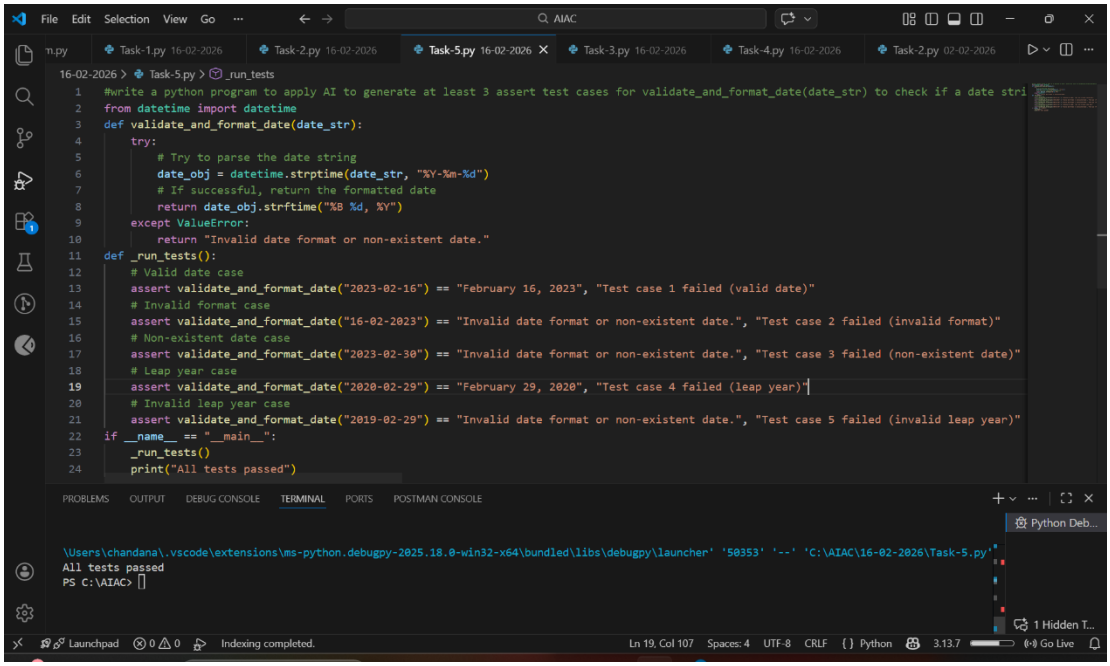
assert validate_and_format_date("02/30/2023") == "Invalid Date"

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:

- Unlike simple string tasks, this requires:
- Checking correct format structure
- Verifying month range (1–12)
- Verifying valid days per month
- Handling leap years (e.g., Feb 29)
- The prompt includes invalid dates like "02/30/2023", which tests logical validation, not just string formatting.
- This ensures the function is logically correct, not superficially correct