ASSIGNMENT-8

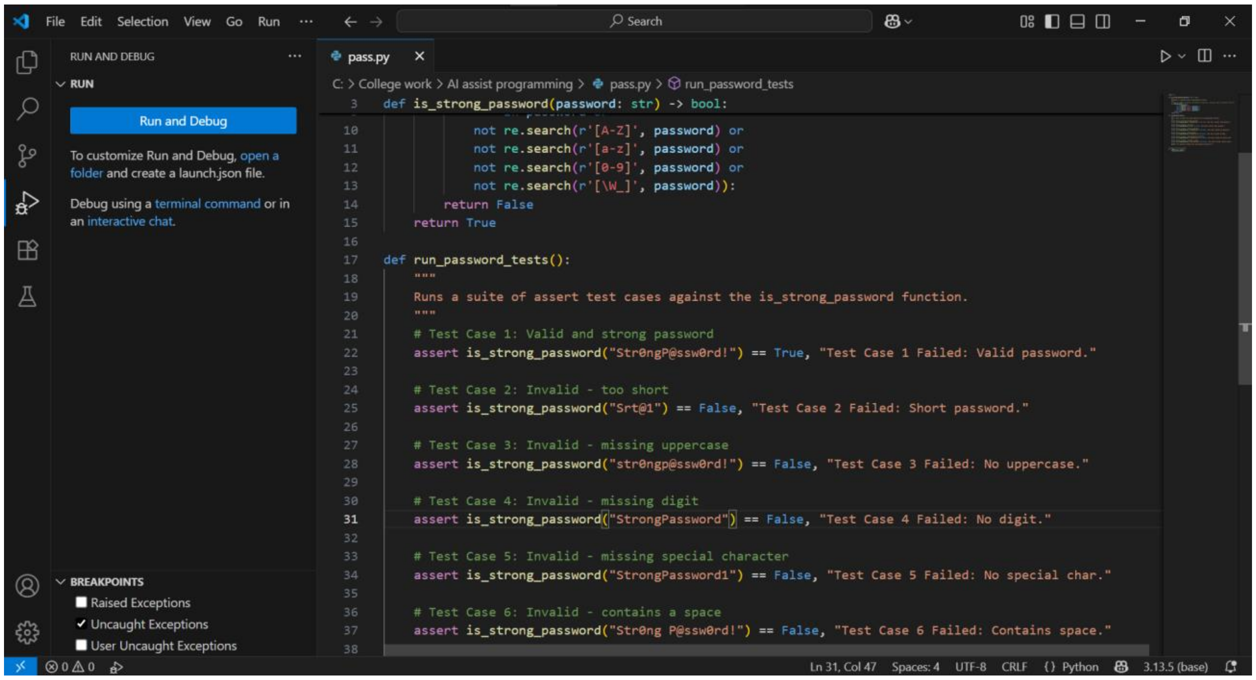
Lab 8: Test-Driven Development with AI – Generating and Working with Test Cases

Lab Objectives:

- To introduce students to test-driven development (TDD) using AI code generation tools.
- To enable the generation of test cases before writing code implementations.
- To reinforce the importance of testing, validation, and error handling.
- To encourage writing clean and reliable code based on AI-generated test expectations

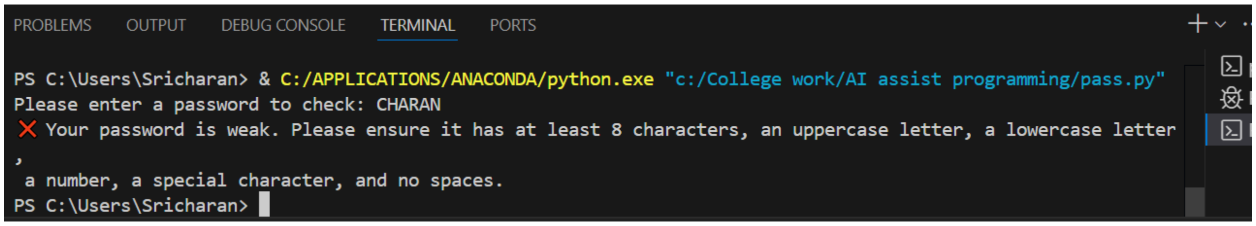
TASK 1: Apply AI to generate at least 3 assert test cases for is_strong_password(password) and implement the validator function.

CODE:



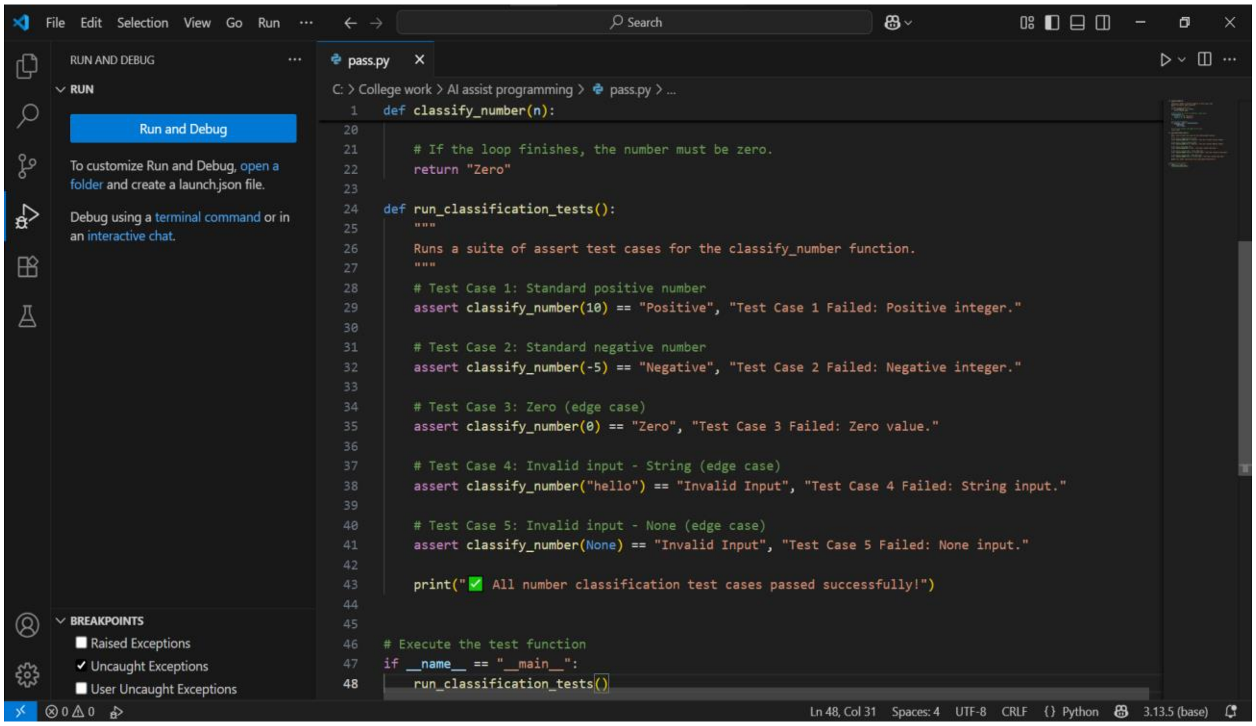
```python
def is_strong_password(password: str) -> bool:
            not re.search(r'[A-Z]', password) or
            not re.search(r'[a-z]', password) or
            not re.search(r'[0-9]', password) or
            not re.search(r'[\W_]', password)):
        return False
    return True

def run_password_tests():
    """
    Runs a suite of assert test cases against the is_strong_password function.
    """
    # Test Case 1: Valid and strong password
    assert is_strong_password("Str0ngP@ssw0rd!") == True, "Test Case 1 Failed: Valid password."

    # Test Case 2: Invalid - too short
    assert is_strong_password("Srt@1") == False, "Test Case 2 Failed: Short password."

    # Test Case 3: Invalid - missing uppercase
    assert is_strong_password("str0ngp@ssw0rd!") == False, "Test Case 3 Failed: No uppercase."

    # Test Case 4: Invalid - missing digit
    assert is_strong_password("StrongPassword") == False, "Test Case 4 Failed: No digit."

    # Test Case 5: Invalid - missing special character
    assert is_strong_password("StrongPassword1") == False, "Test Case 5 Failed: No special char."

    # Test Case 6: Invalid - contains a space
    assert is_strong_password("Str0ng P@ssw0rd!") == False, "Test Case 6 Failed: Contains space."
```

OUTPUT:



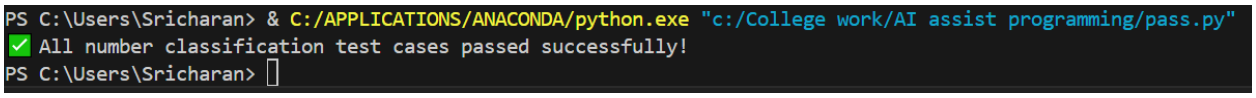
```
PS C:\Users\Sricharan> & C:/APPLICATIONS/ANACONDA/python.exe "c:/College work/AI assist programming/pass.py"
Please enter a password to check: CHARAN
❌ Your password is weak. Please ensure it has at least 8 characters, an uppercase letter, a lowercase letter
,
 a number, a special character, and no spaces.
PS C:\Users\Sricharan>
```

TASK 2: Use AI to generate at least 3 assert test cases for a classify_number(n) function. Implement using loops.



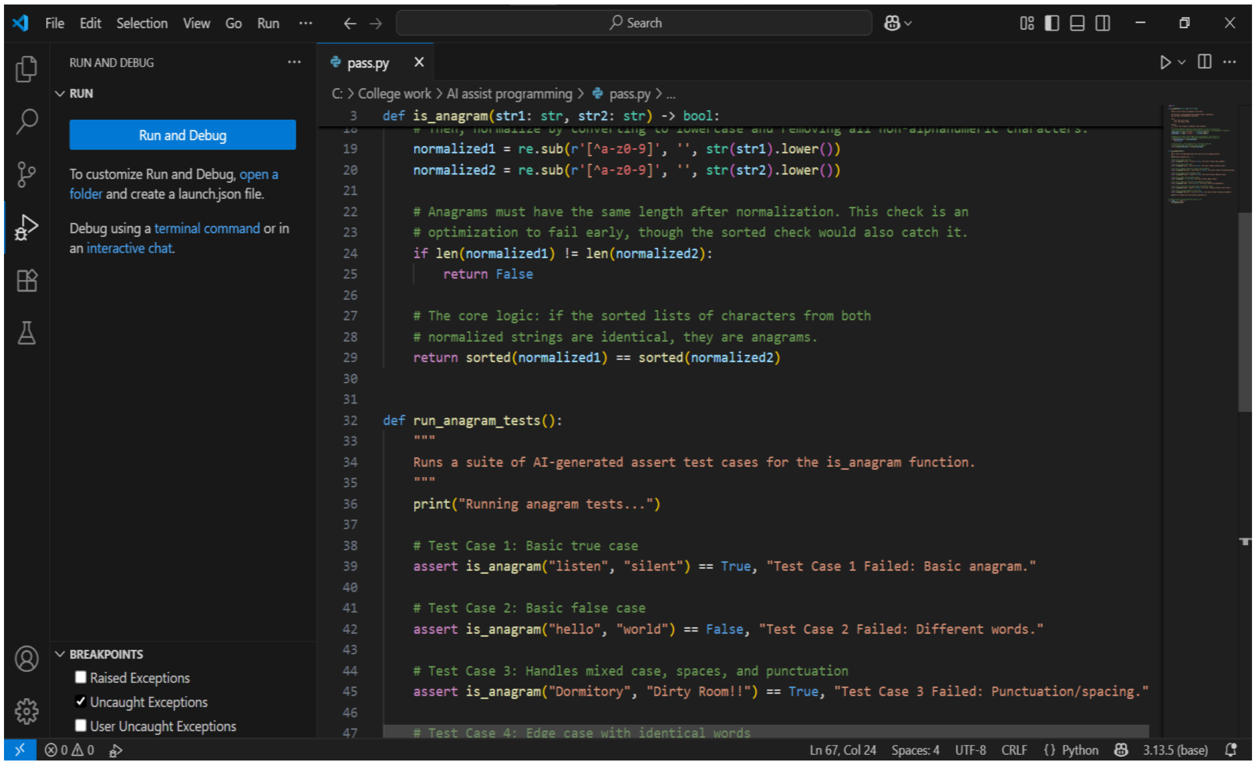
```python
def classify_number(n):

    # If the loop finishes, the number must be zero.
    return "Zero"

def run_classification_tests():
    """
    Runs a suite of assert test cases for the classify_number function.
    """
    # Test Case 1: Standard positive number
    assert classify_number(10) == "Positive", "Test Case 1 Failed: Positive integer."

    # Test Case 2: Standard negative number
    assert classify_number(-5) == "Negative", "Test Case 2 Failed: Negative integer."

    # Test Case 3: Zero (edge case)
    assert classify_number(0) == "Zero", "Test Case 3 Failed: Zero value."

    # Test Case 4: Invalid input - String (edge case)
    assert classify_number("hello") == "Invalid Input", "Test Case 4 Failed: String input."

    # Test Case 5: Invalid input - None (edge case)
    assert classify_number(None) == "Invalid Input", "Test Case 5 Failed: None input."

    print("✅ All number classification test cases passed successfully!")


# Execute the test function
if __name__ == "__main__":
    run_classification_tests()
```

OUTPUT:



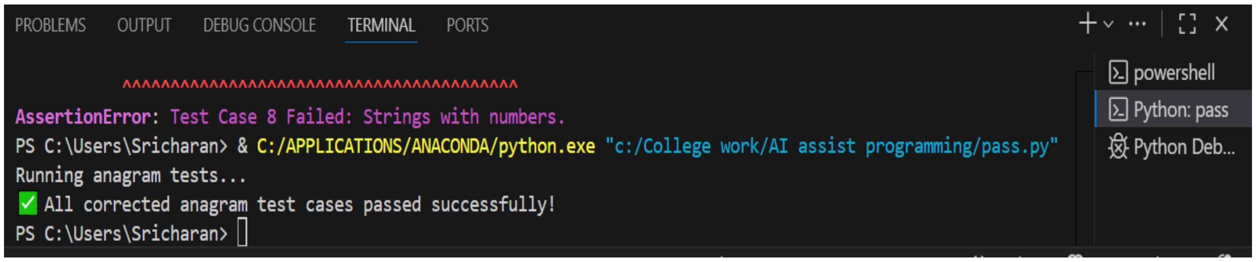
```
PS C:\Users\Sricharan> & C:/APPLICATIONS/ANACONDA/python.exe "c:/College work/AI assist programming/pass.py"
✅ All number classification test cases passed successfully!
PS C:\Users\Sricharan> 
```

TASK 3: Use AI to generate at least 3 assert test cases for is_anagram(str1, str2) and implement the function


File Edit Selection View Go Run
Search
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Debug using a terminal command or in an interactive chat.
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
pass.py
C: > College work > AI assist programming > pass.py > ...
Ln 67, Col 24 Spaces: 4 UTF-8 CRLF {} Python 3.13.5 (base)

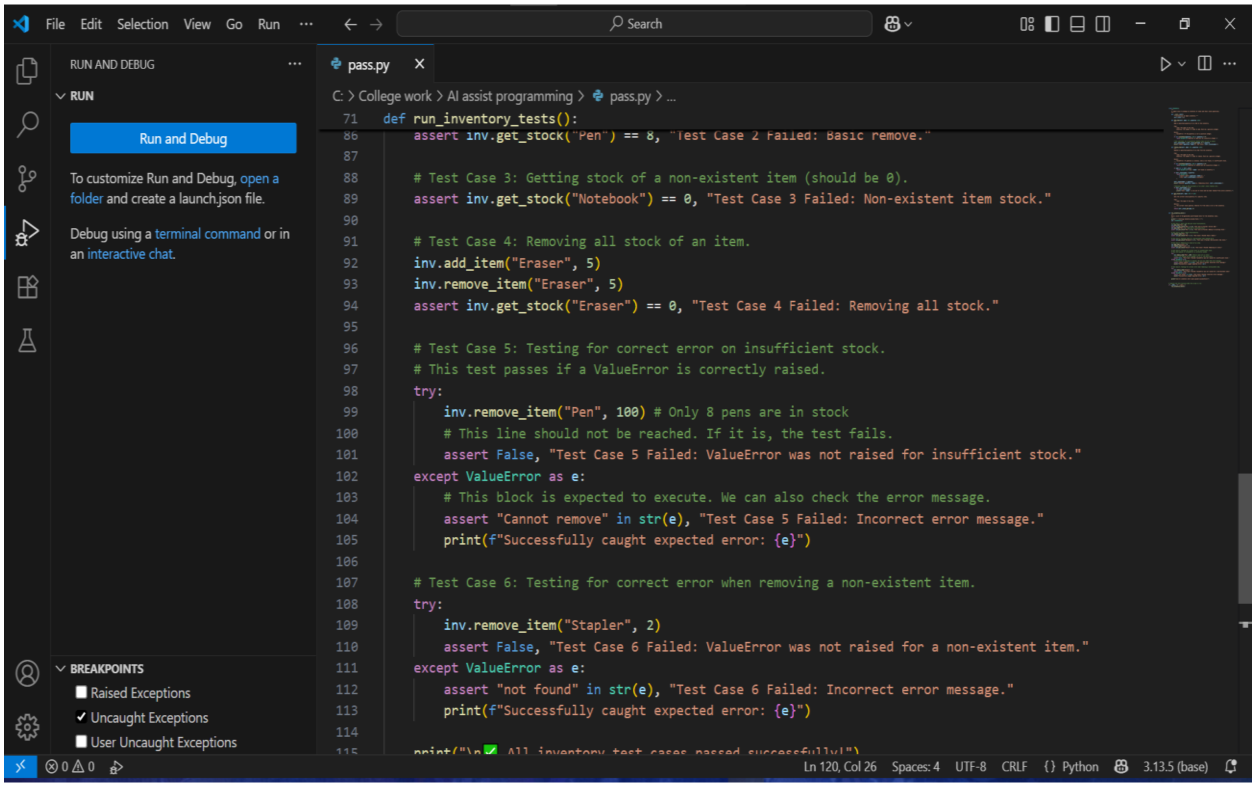
```python
def is_anagram(str1: str, str2: str) -> bool:
    # Then, normalize by converting to lowercase and removing all non-alphanumeric characters.
    normalized1 = re.sub(r'[^a-z0-9]', '', str(str1).lower())
    normalized2 = re.sub(r'[^a-z0-9]', '', str(str2).lower())

    # Anagrams must have the same length after normalization. This check is an
    # optimization to fail early, though the sorted check would also catch it.
    if len(normalized1) != len(normalized2):
        return False

    # The core logic: if the sorted lists of characters from both
    # normalized strings are identical, they are anagrams.
    return sorted(normalized1) == sorted(normalized2)


def run_anagram_tests():
    """
    Runs a suite of AI-generated assert test cases for the is_anagram function.
    """
    print("Running anagram tests...")

    # Test Case 1: Basic true case
    assert is_anagram("listen", "silent") == True, "Test Case 1 Failed: Basic anagram."

    # Test Case 2: Basic false case
    assert is_anagram("hello", "world") == False, "Test Case 2 Failed: Different words."

    # Test Case 3: Handles mixed case, spaces, and punctuation
    assert is_anagram("Dormitory", "Dirty Room!!") == True, "Test Case 3 Failed: Punctuation/spacing."

    # Test Case 4: Edge case with identical words
```

OUTPUT:



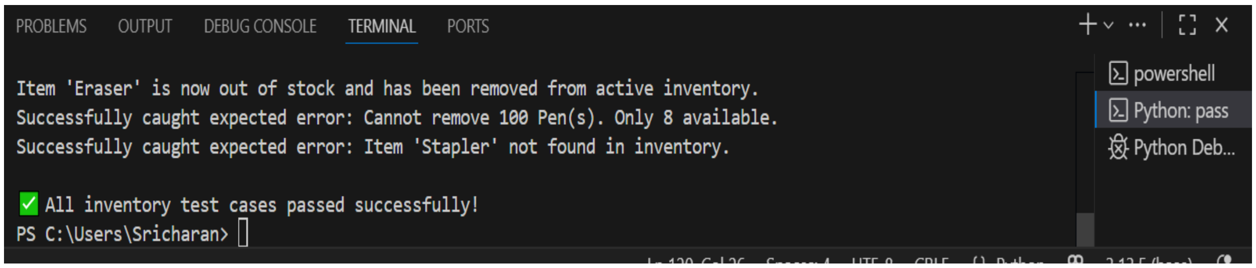
```
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: Test Case 8 Failed: Strings with numbers.
PS C:\Users\Sricharan> & C:/APPLICATIONS/ANACONDA/python.exe "c:/College work/AI assist programming/pass.py"
Running anagram tests...
✅ All corrected anagram test cases passed successfully!
PS C:\Users\Sricharan> 
```

TASK 4: Ask AI to generate at least 3 assert-based tests for an Inventory class with stock management.


File Edit Selection View Go Run
Search
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Debug using a terminal command or in an interactive chat.
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
pass.py
C: > College work > AI assist programming > pass.py > ...
Ln 120, Col 26 Spaces: 4 UTF-8 CRLF {} Python 3.13.5 (base)

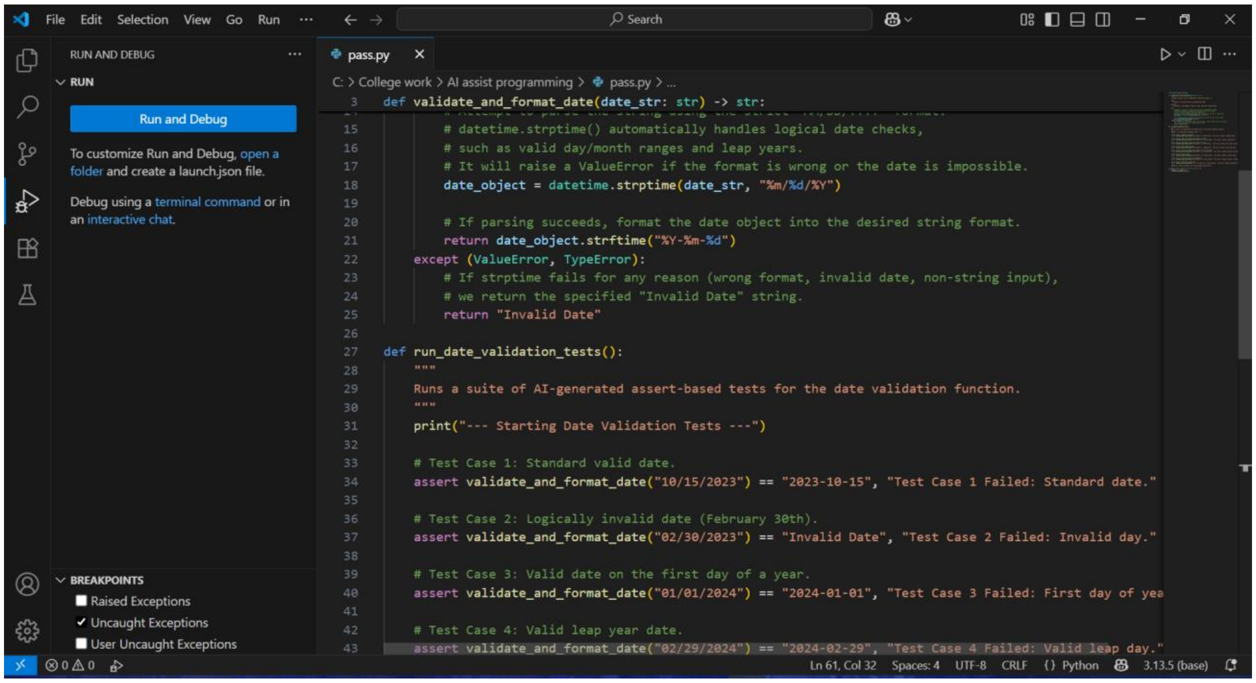
```python
def run_inventory_tests():
    assert inv.get_stock("Pen") == 8, "Test Case 2 Failed: Basic remove."

    # Test Case 3: Getting stock of a non-existent item (should be 0).
    assert inv.get_stock("Notebook") == 0, "Test Case 3 Failed: Non-existent item stock."

    # Test Case 4: Removing all stock of an item.
    inv.add_item("Eraser", 5)
    inv.remove_item("Eraser", 5)
    assert inv.get_stock("Eraser") == 0, "Test Case 4 Failed: Removing all stock."

    # Test Case 5: Testing for correct error on insufficient stock.
    # This test passes if a ValueError is correctly raised.
    try:
        inv.remove_item("Pen", 100) # Only 8 pens are in stock
        # This line should not be reached. If it is, the test fails.
        assert False, "Test Case 5 Failed: ValueError was not raised for insufficient stock."
    except ValueError as e:
        # This block is expected to execute. We can also check the error message.
        assert "Cannot remove" in str(e), "Test Case 5 Failed: Incorrect error message."
        print(f"Successfully caught expected error: {e}")

    # Test Case 6: Testing for correct error when removing a non-existent item.
    try:
        inv.remove_item("Stapler", 2)
        assert False, "Test Case 6 Failed: ValueError was not raised for a non-existent item."
    except ValueError as e:
        assert "not found" in str(e), "Test Case 6 Failed: Incorrect error message."
        print(f"Successfully caught expected error: {e}")

    print("\n✅ All inventory test cases passed successfully!")
```

OUTPUT:



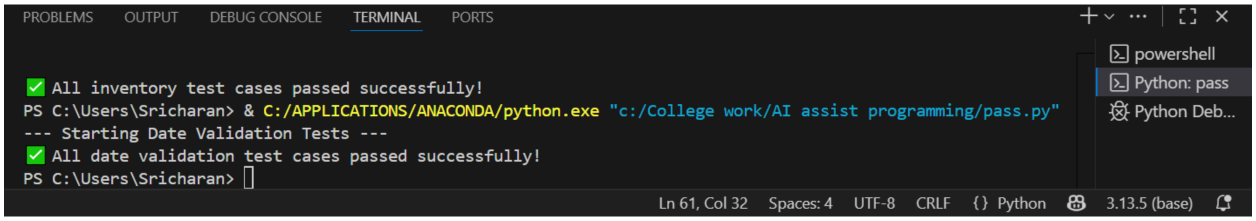
```
Item 'Eraser' is now out of stock and has been removed from active inventory.
Successfully caught expected error: Cannot remove 100 Pen(s). Only 8 available.
Successfully caught expected error: Item 'Stapler' not found in inventory.

✅ All inventory test cases passed successfully!
PS C:\Users\Sricharan>
```

TASK 5: Use AI to generate at least 3 assert test cases for validate_and_format_date(date_str) to check and convert dates.


File Edit Selection View Go Run ···
Search
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Debug using a terminal command or in an interactive chat.
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
pass.py
C: > College work > AI assist programming > pass.py > ...
Ln 61, Col 32 Spaces: 4 UTF-8 CRLF {} Python 3.13.5 (base)

```python
def validate_and_format_date(date_str: str) -> str:
        # datetime.strptime() automatically handles logical date checks,
        # such as valid day/month ranges and leap years.
        # It will raise a ValueError if the format is wrong or the date is impossible.
        date_object = datetime.strptime(date_str, "%m/%d/%Y")

        # If parsing succeeds, format the date object into the desired string format.
        return date_object.strftime("%Y-%m-%d")
    except (ValueError, TypeError):
        # If strptime fails for any reason (wrong format, invalid date, non-string input),
        # we return the specified "Invalid Date" string.
        return "Invalid Date"

def run_date_validation_tests():
    """
    Runs a suite of AI-generated assert-based tests for the date validation function.
    """
    print("--- Starting Date Validation Tests ---")

    # Test Case 1: Standard valid date.
    assert validate_and_format_date("10/15/2023") == "2023-10-15", "Test Case 1 Failed: Standard date."

    # Test Case 2: Logically invalid date (February 30th).
    assert validate_and_format_date("02/30/2023") == "Invalid Date", "Test Case 2 Failed: Invalid day."

    # Test Case 3: Valid date on the first day of a year.
    assert validate_and_format_date("01/01/2024") == "2024-01-01", "Test Case 3 Failed: First day of yea

    # Test Case 4: Valid leap year date.
    assert validate_and_format_date("02/29/2024") == "2024-02-29", "Test Case 4 Failed: Valid leap day."
```

OUTPUT:


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python: pass
Python Deb...
Ln 61, Col 32 Spaces: 4 UTF-8 CRLF {} Python 3.13.5 (base)

```
✅ All inventory test cases passed successfully!
PS C:\Users\Sricharan> & C:/APPLICATIONS/ANACONDA/python.exe "c:/College work/AI assist programming/pass.py"
--- Starting Date Validation Tests ---
✅ All date validation test cases passed successfully!
PS C:\Users\Sricharan>
```