AI Assissted Coding

Lab Assignment- 8.1

Name : G.Abhiram

Ht.No : 2303A51087

Batch No : 02

1. 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:

Password must have at least 8 characters.

Must include uppercase, lowercase, digit, and special character.

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:

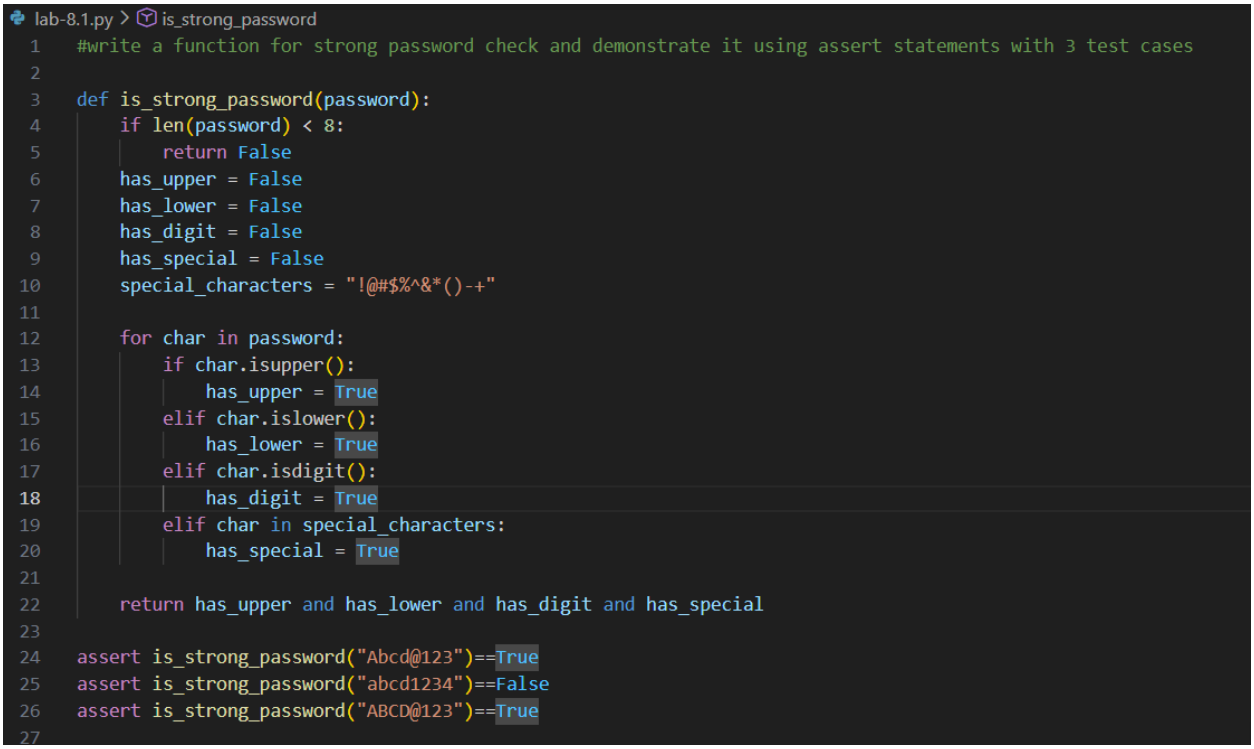
• Password validation logic passing all AI-generated test cases.

```python
#write a function for strong password check and demonstrate it using assert statements with 3 test cases

def is_strong_password(password):
    if len(password) < 8:
        return False
    has_upper = False
    has_lower = False
    has_digit = False
    has_special = False
    special_characters = "!@#$%^&*()-+"

    for char in password:
        if char.isupper():
            has_upper = True
        elif char.islower():
            has_lower = True
        elif char.isdigit():
            has_digit = True
        elif char in special_characters:
            has_special = True

    return has_upper and has_lower and has_digit and has_special
```

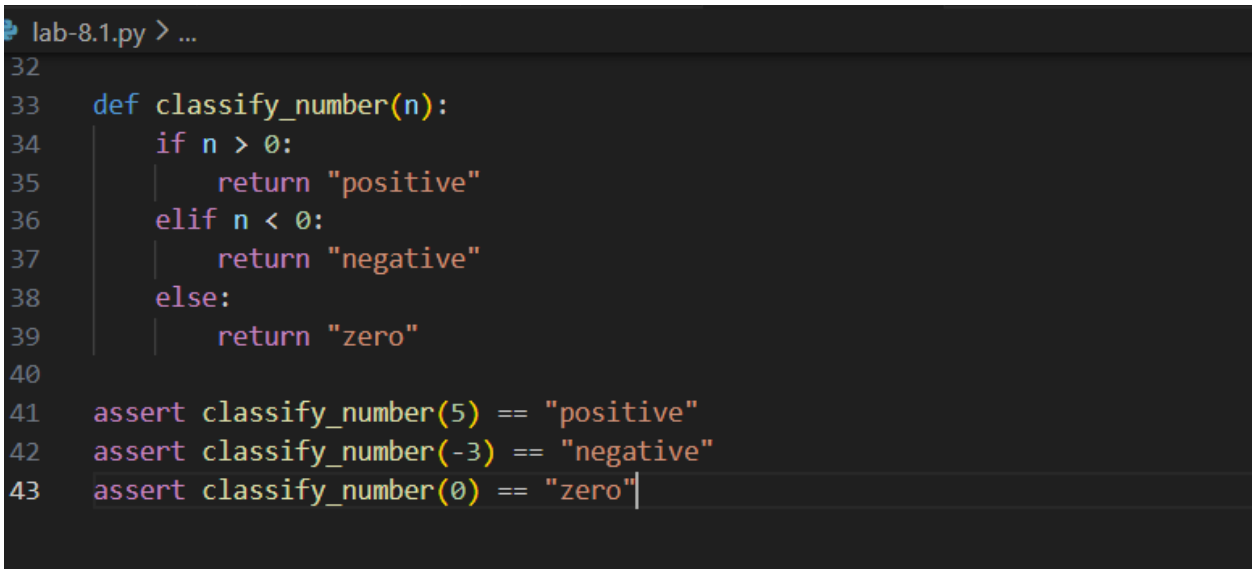


2. Task Description #2 (Number Classification with Loops – Apply AI for Edge Case Handling)

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

• Requirements:

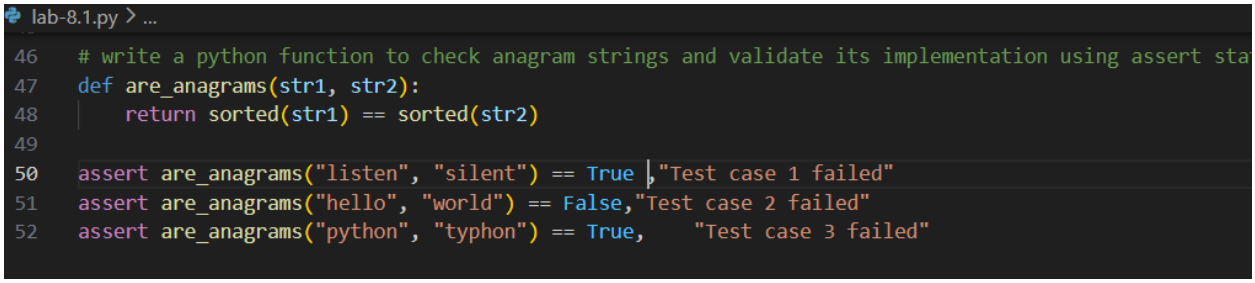
Classify numbers as Positive, Negative, or Zero.
Handle invalid inputs like strings and None.
Include boundary conditions (-1, 0, 1).
Example Assert Test Cases:
assert classify_number(10) == "Positive"
assert classify_number(-5) == "Negative"
assert classify_number(0) == "Zero"
Expected Output #2:
• Classification logic passing all assert tests.



3 . 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:
Ignore case, spaces, and punctuation.
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.



4. 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:

add_item(name, quantity)

remove_item(name, quantity)

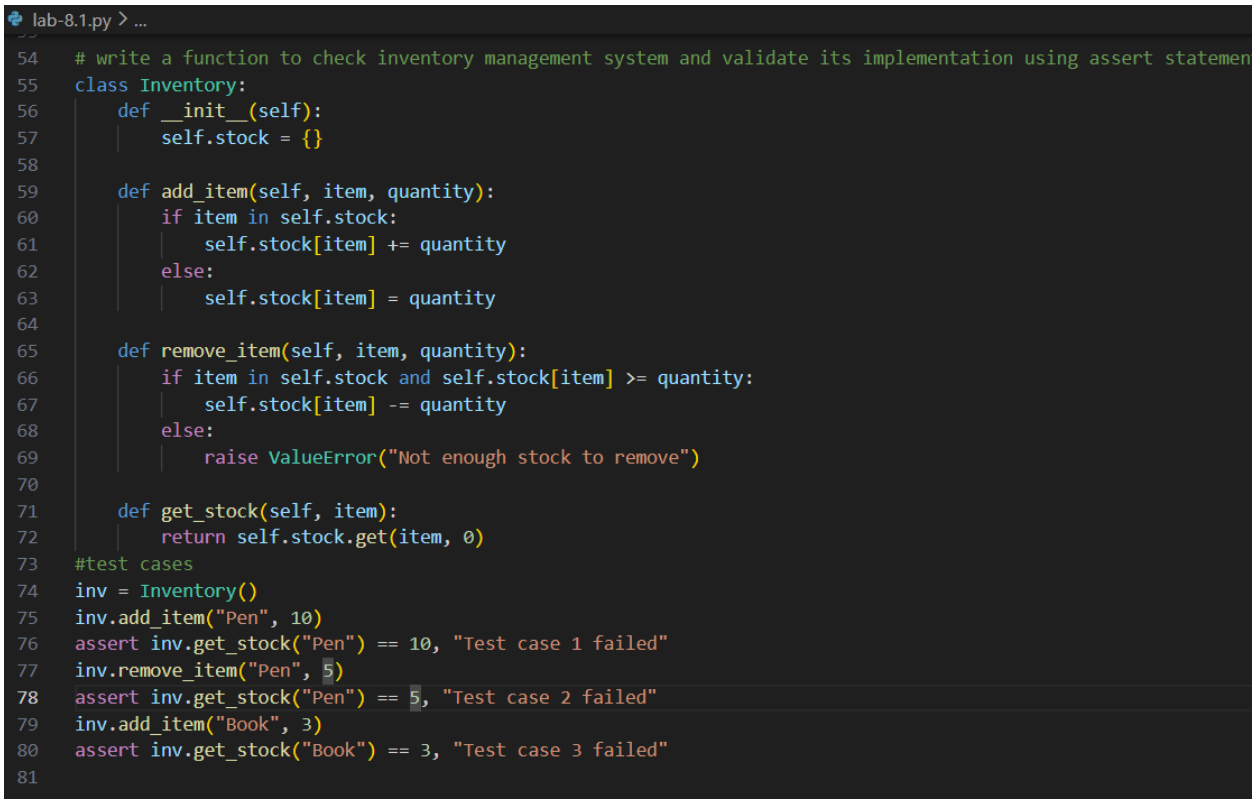
get_stock(name)

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



5. 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:

Validate "MM/DD/YYYY" format.

Handle invalid dates.

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.

lab-8.1.py > ...

```python
# write a function to validate and format date in the format "MM/DD/YYYY" and validate its implementation using assert
from datetime import datetime
def validate_and_format_date(date_str):
    try:
        date_obj = datetime.strptime(date_str, "%m/%d/%Y")
        return date_obj.strftime("%Y-%m-%d")
    except ValueError:
        return "Invalid Date"

# test cases
assert validate_and_format_date("10/15/2023") == "2023-10-15", "Test case 1 failed"
assert validate_and_format_date("02/30/2023") == "Invalid Date",    "Test case 2 failed"
assert validate_and_format_date("01/01/2024") == "2024-01-01", "Test case 3 failed"

```