Name: B. Srikar

Batch : 02 **Course : AI ASSISTED CODING**

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.

Code:

```
def is_strong_password(password):
    if len(password) < 8:
        return False
    if any(char.isspace() for char in password):
        return False
    if not any(char.isupper() for char in password):
        return False
    if not any(char.islower() for char in password):
        return False
    if not any(char.isdigit() for char in password):
        return False
    if not any(char in "!@#$%^&*()-_=+[]{}|;:'\",.<>?/" for char in password):
        return False
    return True
```

Output and test cases:

```
95    assert is_strong_password("Abcd@123") == True
96    assert is_strong_password("abcd123") == False
97    assert is_strong_password("ABCD@1234") == True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
    assert is_strong_password("ABCD@1234") == True
AssertionError
```

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.

• Requirements:

o Classify numbers as Positive, Negative, or Zero.

o Handle invalid inputs like strings and None.

o Include boundary conditions (-1, 0, 1).

Code:

```
def classify_number(n):
    if isinstance(n, (int, float)):
        if n > 0:
            return "Positive"
        elif n < 0:
            return "Negative"
        else:
            return "Zero"
    else:
        return "Invalid Input"
```

Test Case and Output:

```
110    assert classify_number(10) == "Positive"
111    assert classify_number(-5) == "Negative"
112    assert classify_number(0) == "Zero"
113    assert classify_number("string") == "Invalid Input"
114    assert classify_number(None) == "Invalid Input"
115
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS pwsh - AI_ASSISTE

```
PS E:\AI_ASSISTED_CODING> & "c:\Program Files (x86)\Python311-32\python.exe"
5_Assignment_8.1.py
```

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).

Code:

```python
def is_anagram(str1, str2):
    # Remove spaces and punctuation, and convert to lowercase
    def clean_string(s):
        return ''.join(char.lower() for char in s if char.isalnum())

    cleaned_str1 = clean_string(str1)
    cleaned_str2 = clean_string(str2)

    return sorted(cleaned_str1) == sorted(cleaned_str2)
```

Test cases and output:

```
126    assert is_anagram("listen", "silent") == True
127    assert is_anagram("hello", "world") == False
128    assert is_anagram("Dormitory", "Dirty Room") == True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS pwsh - AI_ASSISTED

```
PS E:\AI_ASSISTED_CODING> & "c:\Program Files (x86)\Python311-32\python.exe" e
5_Assignment_8.1.py
```

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)

Code:

```python
class Inventory:
    def __init__(self):
        self.stock = {}

    def add_item(self, name, quantity):
        if name in self.stock:
            self.stock[name] += quantity
        else:
            self.stock[name] = quantity

    def remove_item(self, name, quantity):
        if name in self.stock and self.stock[name] >= quantity:
            self.stock[name] -= quantity
        else:
            raise ValueError("Not enough stock to remove")

    def get_stock(self, name):
        return self.stock.get(name, 0)
```

Test cases and Output:

```python
149    inv = Inventory()
150    inv.add_item("Pen", 10)
151    assert inv.get_stock("Pen") == 10
152    inv.remove_item("Pen", 5)
153    assert inv.get_stock("Pen") == 5
154    inv.add_item("Book", 3)
155    assert inv.get_stock("Book") == 3
156
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  pwsh - AI_ASSISTE

5_Assignment_8.1.py
PS E:\AI_ASSISTED_CODING> & "c:\Program Files (x86)\Python311-32\python.exe"
5_Assignment_8.1.py
```

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".

Code:

```python
def validate_and_format_date(date_str):
    from datetime import datetime

    try:
        date_obj = datetime.strptime(date_str, "%m/%d/%Y")
        return date_obj.strftime("%Y-%m-%d")
    except ValueError:
        return "Invalid Date"
```

Test cases and Output:

```python
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"
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  pwsh - AI_ASSISTED_CODI

5_Assignment_8.1.py
PS E:\AI_ASSISTED_CODING> & "c:\Program Files (x86)\Python311-32\python.exe" e:/AI_
5_Assignment_8.1.py
```