**Assignment – 8.5**

**Name : K. Nithin Kumar**

**Hall Ticket: 2303A51630**

Task Description #1 (Username Validator – Apply AI in Authentication Context)

• Task: Use AI to generate at least 3 assert test cases for a function is_valid_username(username) and then implement the function using Test-Driven Development principles.

• Requirements:

o Username length must be between 5 and 15 characters.

o Must contain only alphabets and digits.

o Must not start with a digit.

o No spaces allowed.

Example Assert Test Cases:

```
assert is_valid_username("User123") == True
assert is_valid_username("12User") == False
assert is_valid_username("Us er") == False
```

Expected Output #1:

• Username validation logic successfully passing all AI-generated test cases

**Code:**

```python
def is_valid_username(username):
    '''
    checks whether a username is valid.

    conditions:
    - length should be between 5 and 15 characters
    - only letters and numbers are allowed
    - should not start with a number
    - spaces are not allowed
    '''
    if len(username) < 5 or len(username) > 15:
        return False
    if not username.isalnum():
        return False
    if username[0].isdigit():
        return False
    return True
# assert test cases
assert is_valid_username("user123") == True
assert is_valid_username("12user") == False
assert is_valid_username("us er") == False
assert is_valid_username("abcd") == False
assert is_valid_username("validuser99") == True
assert is_valid_username("user_123") == False
assert is_valid_username("verylongusername123") == False

print("username validation logic successfully passing all ai-generated test cases.")

```

**Output:**

```
PROBLEMS 44  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  POSTGRESQL QUERY RESULTS  AUGMEN

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITHIN\OneDrive\Desktop\AI
ythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy-2025.18
2' '--' 'C:\Users\NITHIN\OneDrive\Desktop\AI - ASS\8.2.py'
username validation logic successfully passing all ai-generated test cases.
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

## Task Description #2 (Even–Odd & Type Classification – Apply

**AI for Robust Input Handling)**

**• Task: Use AI to generate at least 3 assert test cases for a function classify_value(x) and implement it using conditional logic and loops.**

**• Requirements:**

**o If input is an integer, classify as "Even" or "Odd".**

**o If input is 0, return "Zero".**

**o If input is non-numeric, return "Invalid Input".**

**Example Assert Test Cases:**

```
assert classify_value(8) == "Even"
assert classify_value(7) == "Odd"
assert classify_value("abc") == "Invalid Input"
```

**Expected Output #2:**

**• Function correctly classifying values and passing all test cases.**

## Code:

8.2-2.py > ...

```python
def classify_value(x):
    '''
    classifies the input value.

    rules:
    - if input is integer → even or odd
    - if input is 0 → zero
    - if input is non-numeric → invalid input
    '''
    if isinstance(x, int):

        if x == 0:
            return "Zero"
        for _ in range(1):
            if x % 2 == 0:
                return "Even"
            else:
                return "Odd"
    return "Invalid Input"
# assert test cases
assert classify_value(8) == "Even"
assert classify_value(7) == "Odd"
assert classify_value(0) == "Zero"
assert classify_value("abc") == "Invalid Input"
assert classify_value(4.5) == "Invalid Input"
assert classify_value(-3) == "Odd"

print("function correctly classifying values and passing all test cases.")
```

## Output:

```
PROBLEMS 44  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  POSTGRESQL QUERY RESULTS

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITHIN\OneDrive\Deskt
ythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy-2
8' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\8.2-2.py'
function correctly classifying values and passing all test cases.
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

**Task Description #3 (Palindrome Checker – Apply AI for String Normalization)**

- **Task: Use AI to generate at least 3 assert test cases for a function is_palindrome(text) and implement the function.**
- **Requirements:**
  - **Ignore case, spaces, and punctuation.**
  - **Handle edge cases such as empty strings and single characters.**

**Example Assert Test Cases:**

```
assert is_palindrome("Madam") == True
assert is_palindrome("A man a plan a canal Panama") == True
assert is_palindrome("Python") == False
```

**Expected Output #3:**

- **Function correctly identifying palindromes and passing all AI-generated tests.**

## Code:

```python
def is_palindrome(text):
    '''
    checks whether the given text is a palindrome.

    rules:
    - ignore case, spaces, and punctuation
    - empty string is considered a palindrome
    - single character is also a palindrome
    '''
    number = ""
    for ch in text.lower():
        if ch.isalnum():
            number += ch
    return number == number[::-1]

# assert test cases
assert is_palindrome("Madam") == True
assert is_palindrome("A man a plan a canal Panama") == True
assert is_palindrome("Python") == False
assert is_palindrome("") == True
assert is_palindrome("a") == True
assert is_palindrome("No lemon, no melon!") == True

print("function correctly identifying palindromes and passing all ai-generated tests.")
```

## Output:

```
PROBLEMS 52  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  POSTGRESQL QUERY RESULTS

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITHIN\OneDrive\Des
ythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy
0' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\8.2-3.py'
function correctly identifying palindromes and passing all ai-generated tests.
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

**Task Description #4 (BankAccount Class – Apply AI for**

**Object-Oriented Test-Driven Development)**

- **Task: Ask AI to generate at least 3 assert-based test cases for**

**a BankAccount class and then implement the class.**

- **Methods:**

**o deposit(amount)**

**o withdraw(amount)**

**o get_balance()**

**Example Assert Test Cases:**

```
acc = BankAccount(1000)
acc.deposit(500)
assert acc.get_balance() == 1500
acc.withdraw(300)
assert acc.get_balance() == 1200
```

**Expected Output #4:**

- **Fully functional class that passes all AI-generated assertions.**

## Code:

8.2-4.py > ...

```python
class BankAccount:
    '''
    simple bank account class.
    methods:
    - deposit(amount)
    - withdraw(amount)
    - get_balance()
    '''
    def __init__(self, balance):
        self.balance = balance
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
    def withdraw(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
    def get_balance(self):
        return self.balance

acc = BankAccount(1000)
acc.deposit(500)
assert acc.get_balance() == 1500
acc.withdraw(300)
assert acc.get_balance() == 1200
acc.withdraw(2000)   # should not withdraw (insufficient balance)
assert acc.get_balance() == 1200
acc.deposit(-100)    # invalid deposit
assert acc.get_balance() == 1200
print("fully functional class that passes all ai-generated assertions.")

```

## Output:

```
PROBLEMS 57  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  POSTGRESQL QUERY RE

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users\NITHIN\OneDriv
ythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.de
7' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\8.2-4.py'
fully functional class that passes all ai-generated assertions.
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

**Task Description #5 (Email ID Validation – Apply AI for Data Validation)**

- **Task: Use AI to generate at least 3 assert test cases for a function validate_email(email) and implement the function.**
- **Requirements:**
  - **Must contain @ and .**
  - **Must not start or end with special characters.**
  - **Should handle invalid formats gracefully.**

**Example Assert Test Cases:**

```
assert validate_email("user@example.com") == True
assert validate_email("userexample.com") == False
assert validate_email("@gmail.com") == False
```

**Expected Output #5:**

- **Email validation function passing all AI-generated test cases and handling edge cases correctly.**

## Code:

8.2-5.py > validate_email

```python
def validate_email(email):
    '''
    checks whether the given email is valid.
    rules:
    - must contain @ and .
    - should not start or end with special characters
    - should handle invalid formats safely
    '''
    if not isinstance(email, str):
        return False
    if "@" not in email or "." not in email:
        return False
    if not email[0].isalnum() or not email[-1].isalnum():
        return False
    parts = email.split("@")
    if len(parts) != 2:
        return False
    local, domain = parts
    if local == "" or domain == "":
        return False
    if "." not in domain:
        return False
    return True
# assert test cases
assert validate_email("user@example.com") == True
assert validate_email("userexample.com") == False
assert validate_email("@gmail.com") == False
assert validate_email("user@.com") == False
assert validate_email(".user@gmail.com") == False
assert validate_email("user@gmail.com.") == False
assert validate_email("user123@test.co") == True

print("email validation function passing all ai-generated test cases.")
```

## Output: