AI Assisted Coding

Assignment – 8.2

P.Manikanta || 2303A51271 || Batch:- 8

Task 1 – Test-Driven Development for Even/Odd Number Validator

• Use AI tools to first generate test cases for a function is_even(n)

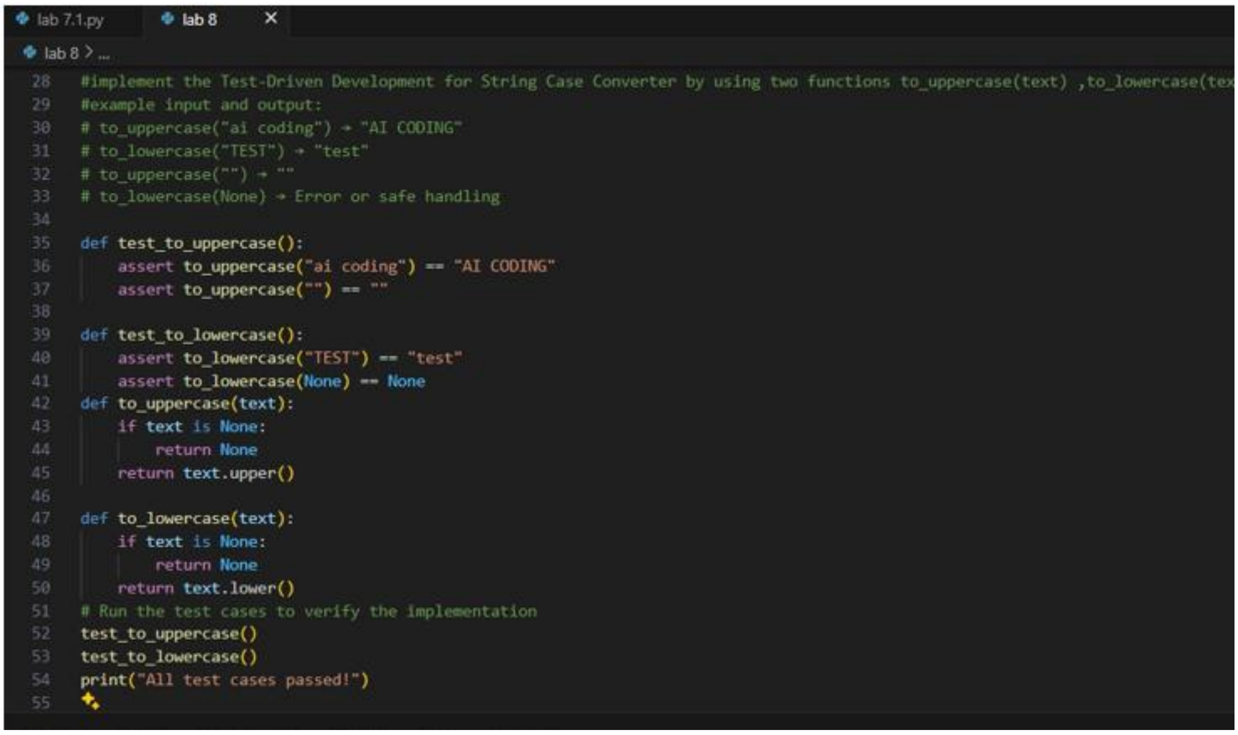
and then implement the function so that it satisfies all generated

tests.

Requirements:

• Input must be an integer

• Handle zero, negative numbers, and large integers



```python
#implement the Test-Driven Development for String Case Converter by using two functions to_uppercase(text) ,to_lowercase(tex
#example input and output:
# to_uppercase("ai coding") → "AI CODING"
# to_lowercase("TEST") → "test"
# to_uppercase("") → ""
# to_lowercase(None) → Error or safe handling

def test_to_uppercase():
    assert to_uppercase("ai coding") == "AI CODING"
    assert to_uppercase("") == ""

def test_to_lowercase():
    assert to_lowercase("TEST") == "test"
    assert to_lowercase(None) == None
def to_uppercase(text):
    if text is None:
        return None
    return text.upper()

def to_lowercase(text):
    if text is None:
        return None
    return text.lower()
# Run the test cases to verify the implementation
test_to_uppercase()
test_to_lowercase()
print("All test cases passed!")
```

Task 2 – Test-Driven Development for String Case Converter

• Ask AI to generate test cases for two functions:

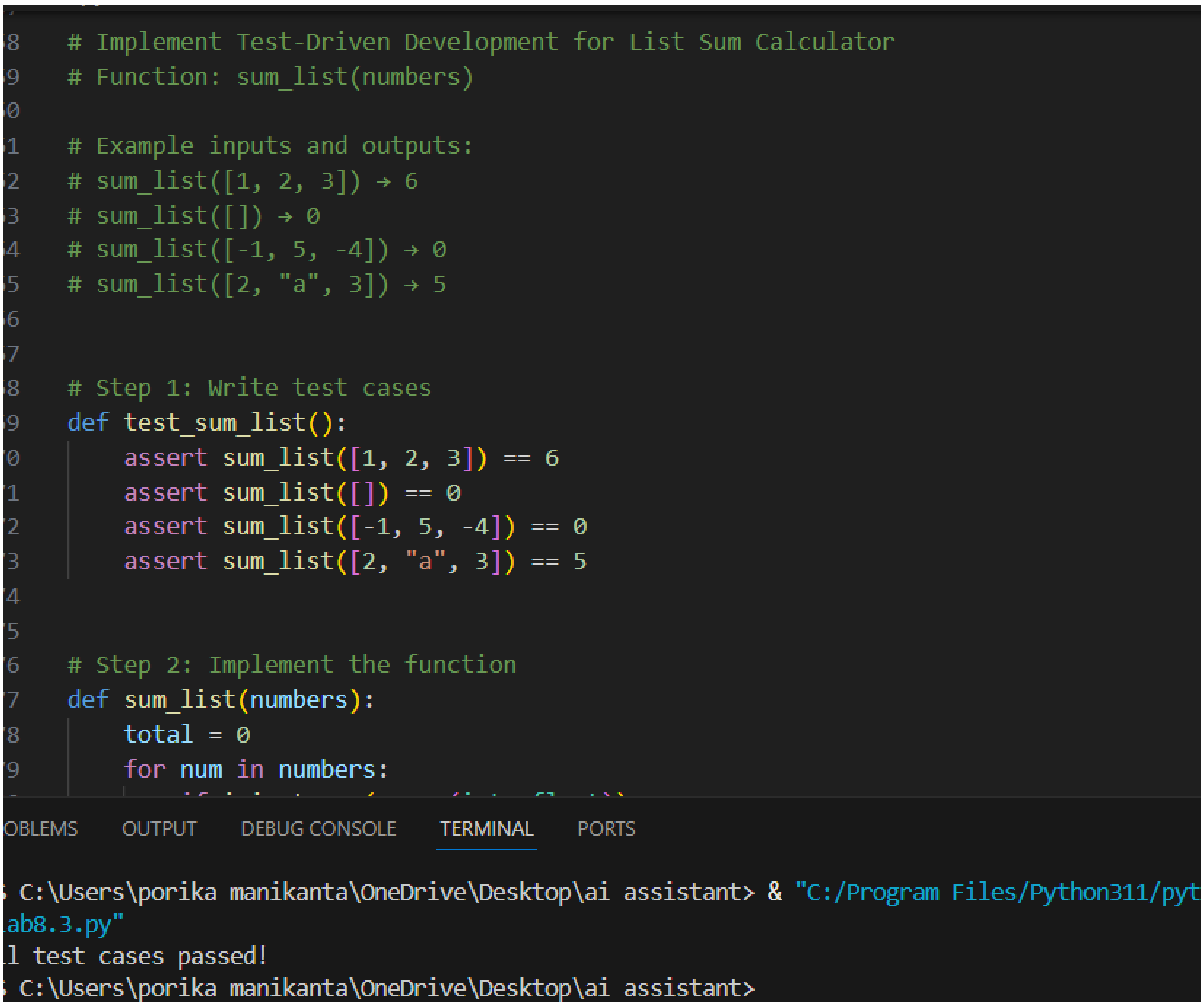
• to_uppercase(text)

• to_lowercase(text) Requirements:

• Handle empty strings

• Handle mixed-case input

• Handle invalid inputs such as numbers or None



```python
# Implement Test-Driven Development for List Sum Calculator
# Function: sum_list(numbers)

# Example inputs and outputs:
# sum_list([1, 2, 3]) → 6
# sum_list([]) → 0
# sum_list([-1, 5, -4]) → 0
# sum_list([2, "a", 3]) → 5


# Step 1: Write test cases
def test_sum_list():
    assert sum_list([1, 2, 3]) == 6
    assert sum_list([]) == 0
    assert sum_list([-1, 5, -4]) == 0
    assert sum_list([2, "a", 3]) == 5


# Step 2: Implement the function
def sum_list(numbers):
    total = 0
    for num in numbers:
```

```
 C:\Users\porika manikanta\OneDrive\Desktop\ai assistant> & "C:/Program Files/Python311/pyt
ab8.3.py"
l test cases passed!
 C:\Users\porika manikanta\OneDrive\Desktop\ai assistant>
```

Task 3 – Test-Driven Development for List Sum Calculator

• Use AI to generate test cases for a function sum_list(numbers) that

  calculates the sum of list elements.

Requirements:

• Handle empty lists

• Handle negative numbers

• Ignore or safely handle non-numeric values

Task 4 – Test Cases for Student Result Class

• Generate test cases for a StudentResult class with the following methods:

• add_marks(mark)

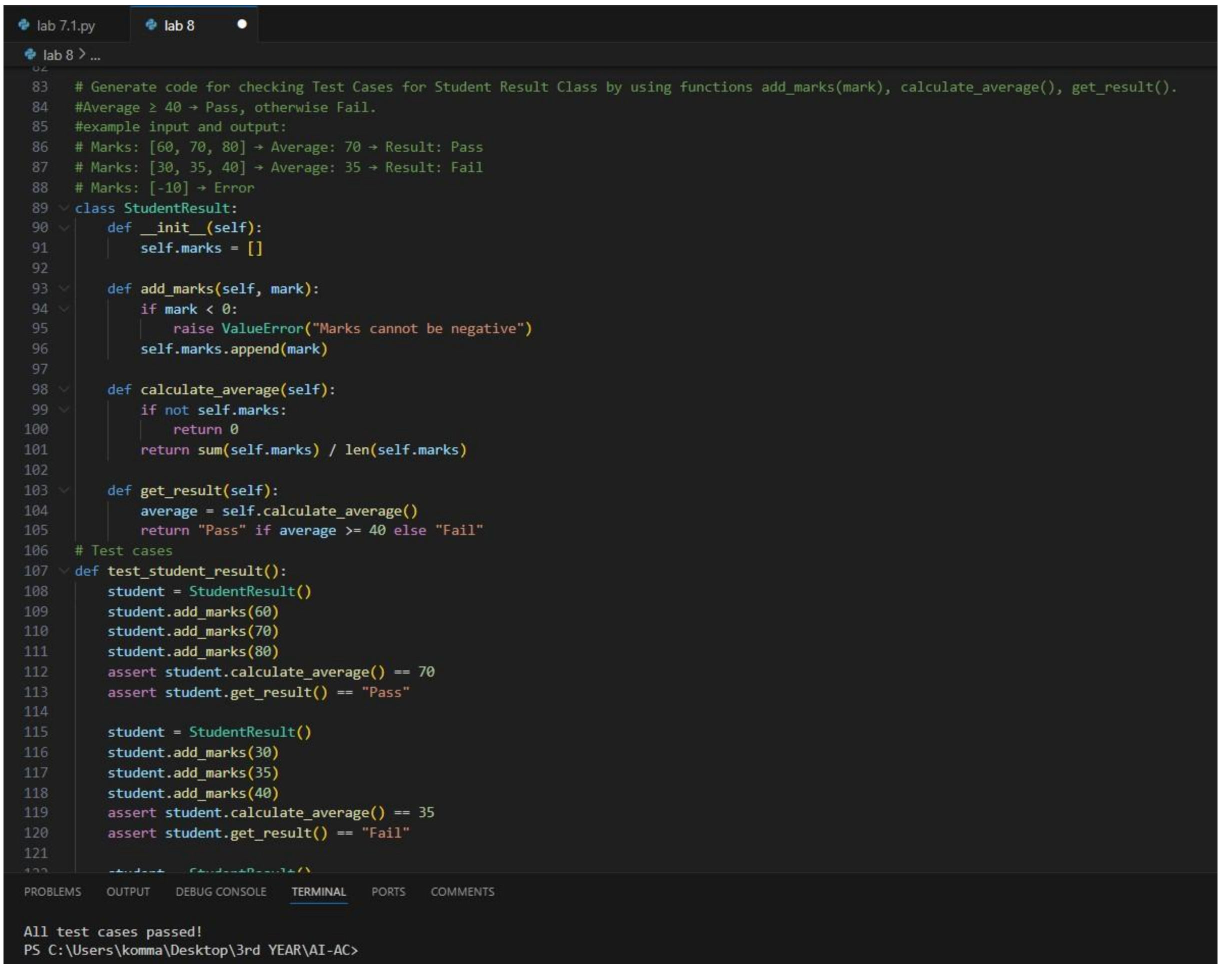
• calculate_average()

• get_result()

Requirements:

• Marks must be between 0 and 100

• Average ≥ 40 → Pass, otherwise Fail



```python
# Generate code for checking Test Cases for Student Result Class by using functions add_marks(mark), calculate_average(), get_result().
#Average ≥ 40 → Pass, otherwise Fail.
#example input and output:
# Marks: [60, 70, 80] → Average: 70 → Result: Pass
# Marks: [30, 35, 40] → Average: 35 → Result: Fail
# Marks: [-10] → Error
class StudentResult:
    def __init__(self):
        self.marks = []

    def add_marks(self, mark):
        if mark < 0:
            raise ValueError("Marks cannot be negative")
        self.marks.append(mark)

    def calculate_average(self):
        if not self.marks:
            return 0
        return sum(self.marks) / len(self.marks)

    def get_result(self):
        average = self.calculate_average()
        return "Pass" if average >= 40 else "Fail"
# Test cases
def test_student_result():
    student = StudentResult()
    student.add_marks(60)
    student.add_marks(70)
    student.add_marks(80)
    assert student.calculate_average() == 70
    assert student.get_result() == "Pass"

    student = StudentResult()
    student.add_marks(30)
    student.add_marks(35)
    student.add_marks(40)
    assert student.calculate_average() == 35
    assert student.get_result() == "Fail"

```

```
All test cases passed!
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```

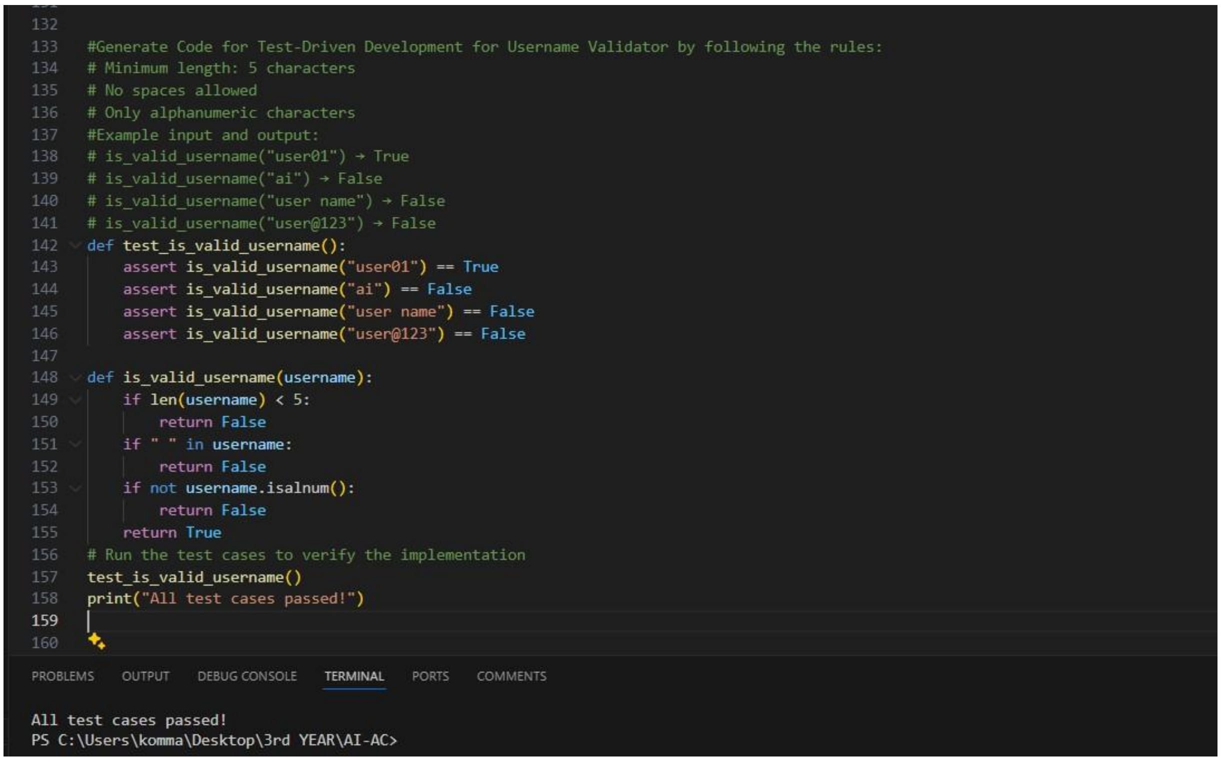
Task 5 – Test-Driven Development for Username Validator

Requirements:

• Minimum length: 5 characters

• No spaces allowed

• Only alphanumeric characters



```python

#Generate Code for Test-Driven Development for Username Validator by following the rules:
# Minimum length: 5 characters
# No spaces allowed
# Only alphanumeric characters
#Example input and output:
# is_valid_username("user01") → True
# is_valid_username("ai") → False
# is_valid_username("user name") → False
# is_valid_username("user@123") → False
def test_is_valid_username():
    assert is_valid_username("user01") == True
    assert is_valid_username("ai") == False
    assert is_valid_username("user name") == False
    assert is_valid_username("user@123") == False

def is_valid_username(username):
    if len(username) < 5:
        return False
    if " " in username:
        return False
    if not username.isalnum():
        return False
    return True
# Run the test cases to verify the implementation
test_is_valid_username()
print("All test cases passed!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
All test cases passed!
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```