AI Assisted Coding

Assignment – 8.2

M.Sathwik || 2303A51483 || Batch:- 08

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

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

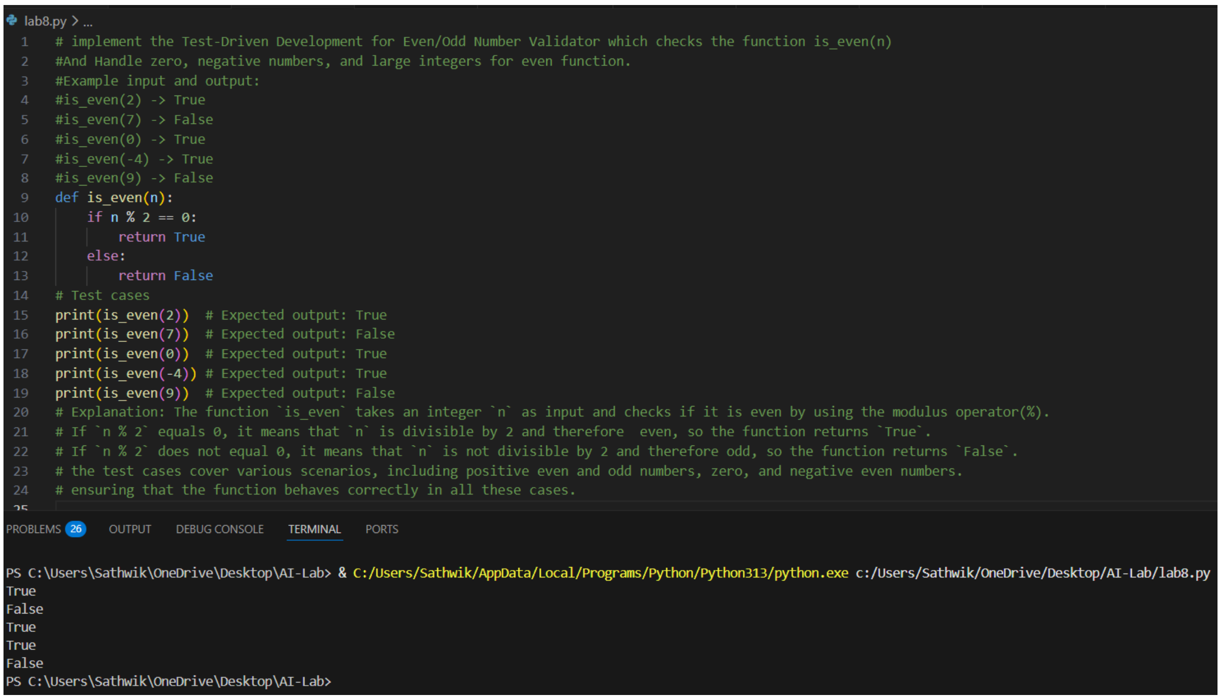
and then implement the function so that it satisfies all generated

tests.

Requirements:

• Input must be an integer

• Handle zero, negative numbers, and large integers



```python
# implement the Test-Driven Development for Even/Odd Number Validator which checks the function is_even(n)
#And Handle zero, negative numbers, and large integers for even function.
#Example input and output:
#is_even(2) -> True
#is_even(7) -> False
#is_even(0) -> True
#is_even(-4) -> True
#is_even(9) -> False
def is_even(n):
    if n % 2 == 0:
        return True
    else:
        return False
# Test cases
print(is_even(2))  # Expected output: True
print(is_even(7))  # Expected output: False
print(is_even(0))  # Expected output: True
print(is_even(-4)) # Expected output: True
print(is_even(9))  # Expected output: False
# Explanation: The function `is_even` takes an integer `n` as input and checks if it is even by using the modulus operator(%).
# If `n % 2` equals 0, it means that `n` is divisible by 2 and therefore  even, so the function returns `True`.
# If `n % 2` does not equal 0, it means that `n` is not divisible by 2 and therefore odd, so the function returns `False`.
# the test cases cover various scenarios, including positive even and odd numbers, zero, and negative even numbers.
# ensuring that the function behaves correctly in all these cases.
```

```
PS C:\Users\Sathwik\OneDrive\Desktop\AI-Lab> & C:/Users/Sathwik/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Sathwik/OneDrive/Desktop/AI-Lab/lab8.py
True
False
True
True
False
PS C:\Users\Sathwik\OneDrive\Desktop\AI-Lab>
```

Task 2 – Test-Driven Development for String Case Converter

• Ask AI to generate test cases for two functions:

• to_uppercase(text)

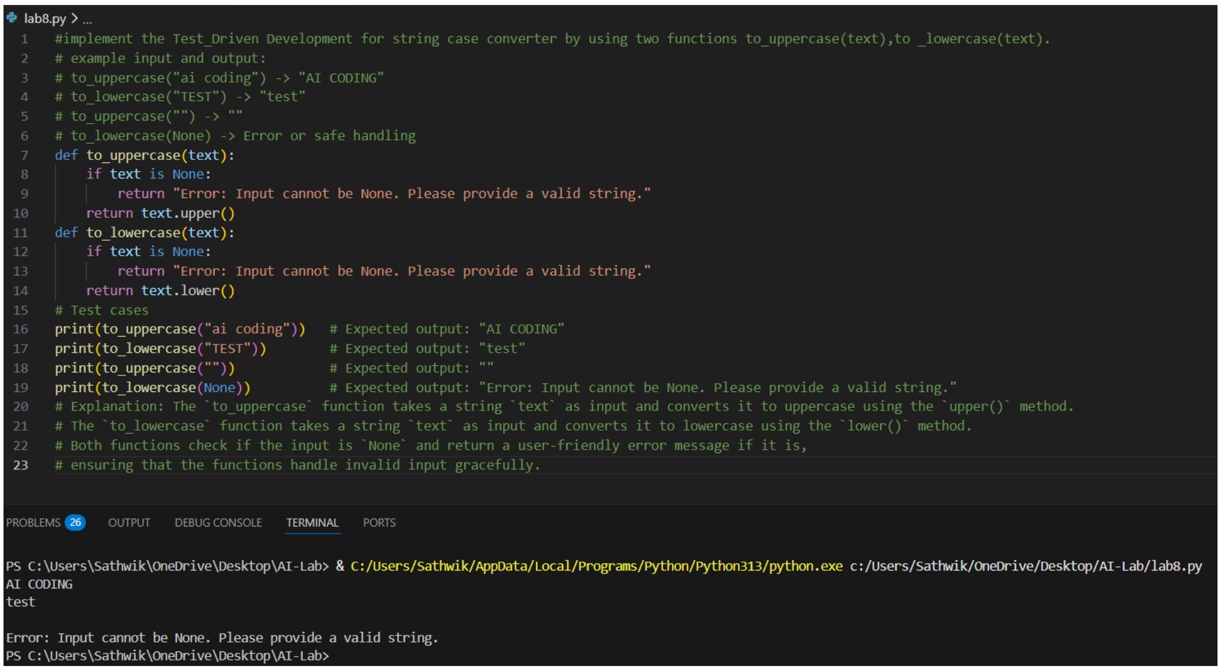
• to_lowercase(text)

Requirements:

• Handle empty strings

• Handle mixed-case input

• Handle invalid inputs such as numbers or None



lab8.py > ...

```python
#implement the Test_Driven Development for string case converter by using two functions to_uppercase(text),to _lowercase(text).
# example input and output:
# to_uppercase("ai coding") -> "AI CODING"
# to_lowercase("TEST") -> "test"
# to_uppercase("") -> ""
# to_lowercase(None) -> Error or safe handling
def to_uppercase(text):
    if text is None:
        return "Error: Input cannot be None. Please provide a valid string."
    return text.upper()
def to_lowercase(text):
    if text is None:
        return "Error: Input cannot be None. Please provide a valid string."
    return text.lower()
# Test cases
print(to_uppercase("ai coding"))   # Expected output: "AI CODING"
print(to_lowercase("TEST"))        # Expected output: "test"
print(to_uppercase(""))            # Expected output: ""
print(to_lowercase(None))          # Expected output: "Error: Input cannot be None. Please provide a valid string."
# Explanation: The `to_uppercase` function takes a string `text` as input and converts it to uppercase using the `upper()` method.
# The `to_lowercase` function takes a string `text` as input and converts it to lowercase using the `lower()` method.
# Both functions check if the input is `None` and return a user-friendly error message if it is,
# ensuring that the functions handle invalid input gracefully.
```

```
PS C:\Users\Sathwik\OneDrive\Desktop\AI-Lab> & C:/Users/Sathwik/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Sathwik/OneDrive/Desktop/AI-Lab/lab8.py
AI CODING
test

Error: Input cannot be None. Please provide a valid string.
PS C:\Users\Sathwik\OneDrive\Desktop\AI-Lab>
```

Task 3 – Test-Driven Development for List Sum Calculator

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

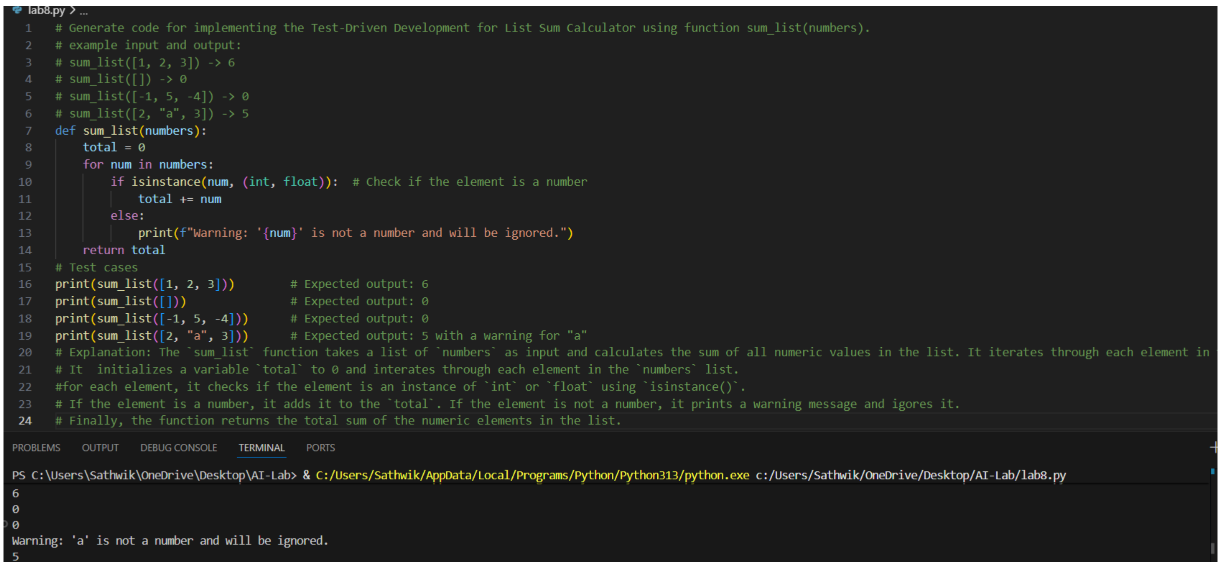
that calculates the sum of list elements.

Requirements:

• Handle empty lists

• Handle negative numbers

• Ignore or safely handle non-numeric values



lab8.py > ...

```python
# Generate code for implementing the Test-Driven Development for List Sum Calculator using function sum_list(numbers).
# example input and output:
# sum_list([1, 2, 3]) -> 6
# sum_list([]) -> 0
# sum_list([-1, 5, -4]) -> 0
# sum_list([2, "a", 3]) -> 5
def sum_list(numbers):
    total = 0
    for num in numbers:
        if isinstance(num, (int, float)):  # Check if the element is a number
            total += num
        else:
            print(f"Warning: '{num}' is not a number and will be ignored.")
    return total
# Test cases
print(sum_list([1, 2, 3]))        # Expected output: 6
print(sum_list([]))               # Expected output: 0
print(sum_list([-1, 5, -4]))      # Expected output: 0
print(sum_list([2, "a", 3]))      # Expected output: 5 with a warning for "a"
# Explanation: The `sum_list` function takes a list of `numbers` as input and calculates the sum of all numeric values in the list. It iterates through each element in
# It  initializes a variable `total` to 0 and interates through each element in the `numbers` list.
#for each element, it checks if the element is an instance of `int` or `float` using `isinstance()`.
# If the element is a number, it adds it to the `total`. If the element is not a number, it prints a warning message and igores it.
# Finally, the function returns the total sum of the numeric elements in the list.
```

```
PS C:\Users\Sathwik\OneDrive\Desktop\AI-Lab> & C:/Users/Sathwik/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Sathwik/OneDrive/Desktop/AI-Lab/lab8.py
6
0
0
Warning: 'a' is not a number and will be ignored.
5
```

Task 4 – Test Cases for Student Result Class

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

• add_marks(mark)

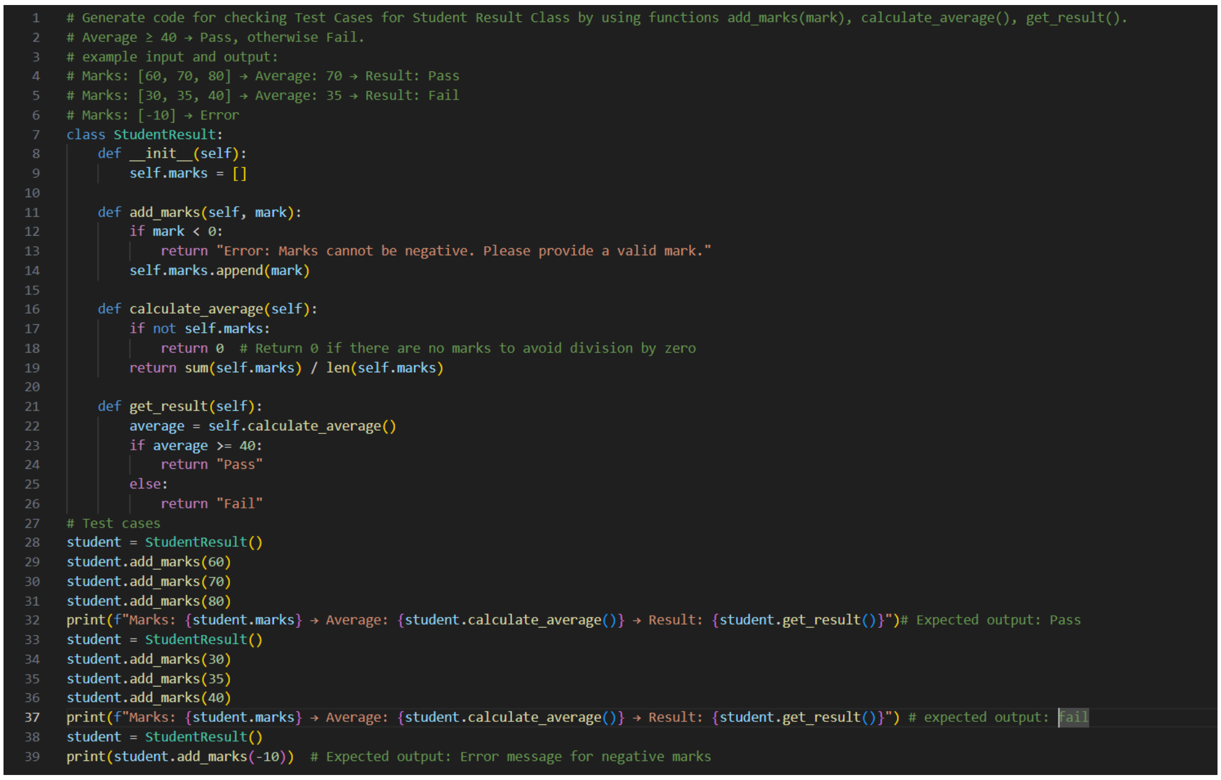
• calculate_average()

• get_result()

Requirements:

• Marks must be between 0 and 100

• Average ≥ 40 → Pass, otherwise Fail

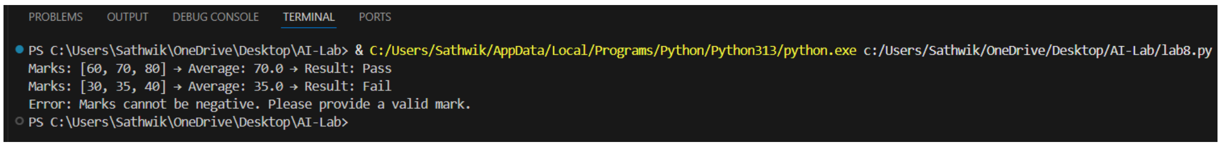


```python
# Generate code for checking Test Cases for Student Result Class by using functions add_marks(mark), calculate_average(), get_result().
# Average ≥ 40 → Pass, otherwise Fail.
# example input and output:
# Marks: [60, 70, 80] → Average: 70 → Result: Pass
# Marks: [30, 35, 40] → Average: 35 →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

    def add_marks(self, mark):
        if mark < 0:
            return "Error: Marks cannot be negative. Please provide a valid mark."
        self.marks.append(mark)

    def calculate_average(self):
        if not self.marks:
            return 0  # Return 0 if there are no marks to avoid division by zero
        return sum(self.marks) / len(self.marks)

    def get_result(self):
        average = self.calculate_average()
        if average >= 40:
            return "Pass"
        else:
            return "Fail"
# Test cases
student = StudentResult()
student.add_marks(60)
student.add_marks(70)
student.add_marks(80)
print(f"Marks: {student.marks} → Average: {student.calculate_average()} → Result: {student.get_result()}")# Expected output: Pass
student = StudentResult()
student.add_marks(30)
student.add_marks(35)
student.add_marks(40)
print(f"Marks: {student.marks} → Average: {student.calculate_average()} → Result: {student.get_result()}") # expected output: Fail
student = StudentResult()
print(student.add_marks(-10))  # Expected output: Error message for negative marks
```



PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\Sathwik\OneDrive\Desktop\AI-Lab> & C:/Users/Sathwik/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Sathwik/OneDrive/Desktop/AI-Lab/lab8.py
Marks: [60, 70, 80] → Average: 70.0 → Result: Pass
Marks: [30, 35, 40] → Average: 35.0 → Result: Fail
Error: Marks cannot be negative. Please provide a valid mark.
PS C:\Users\Sathwik\OneDrive\Desktop\AI-Lab>
```

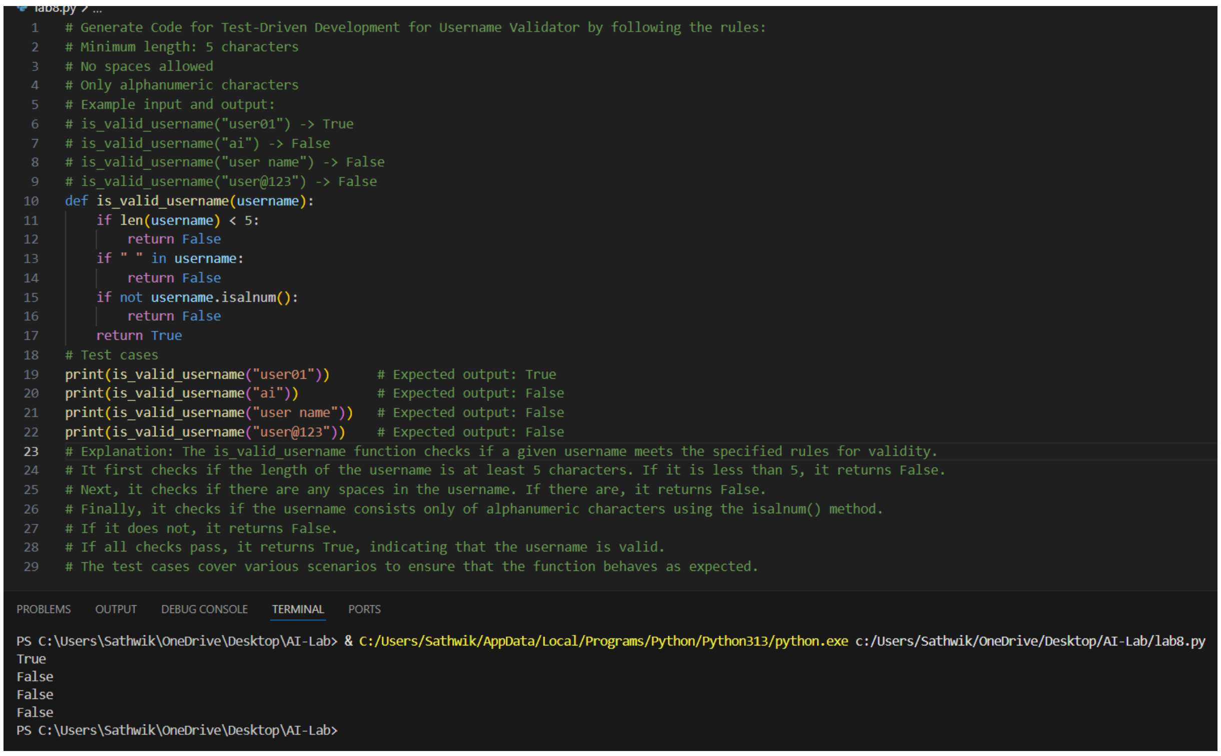
Task 5 – Test-Driven Development for Username Validator

Requirements:

• Minimum length: 5 characters

• No spaces allowed

• Only alphanumeric characters



lab8.py > ...

```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 is_valid_username(username):
    if len(username) < 5:
        return False
    if " " in username:
        return False
    if not username.isalnum():
        return False
    return True
# Test cases
print(is_valid_username("user01"))      # Expected output: True
print(is_valid_username("ai"))          # Expected output: False
print(is_valid_username("user name"))   # Expected output: False
print(is_valid_username("user@123"))    # Expected output: False
# Explanation: The is_valid_username function checks if a given username meets the specified rules for validity.
# It first checks if the length of the username is at least 5 characters. If it is less than 5, it returns False.
# Next, it checks if there are any spaces in the username. If there are, it returns False.
# Finally, it checks if the username consists only of alphanumeric characters using the isalnum() method.
# If it does not, it returns False.
# If all checks pass, it returns True, indicating that the username is valid.
# The test cases cover various scenarios to ensure that the function behaves as expected.
```

```
PS C:\Users\Sathwik\OneDrive\Desktop\AI-Lab> & C:/Users/Sathwik/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Sathwik/OneDrive/Desktop/AI-Lab/lab8.py
True
False
False
False
PS C:\Users\Sathwik\OneDrive\Desktop\AI-Lab>
```