ASSIGNMENT-8.2

Name: B.Shravya

HT. No: 2303A51492

Batch: 08

Lab 8: Test-Driven Development with AI – Generating and Working with Test Cases

Task Description

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

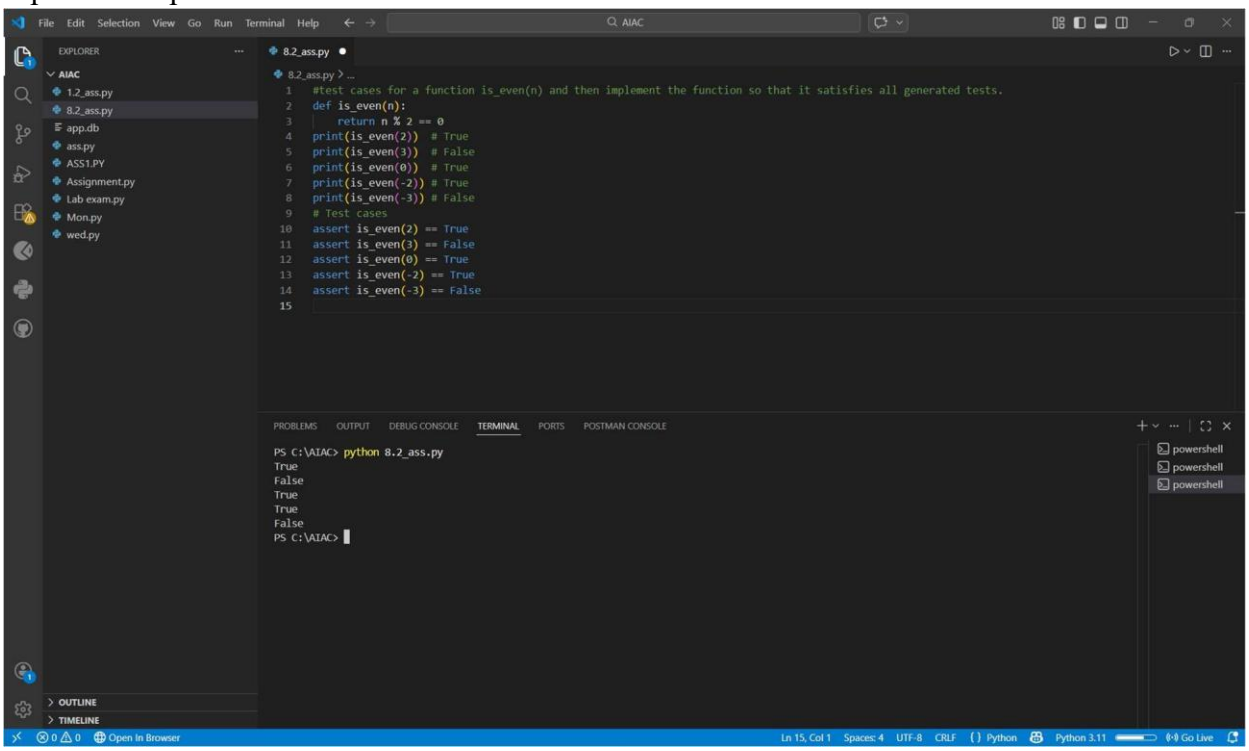
• Handle zero, negative numbers, and large integers Example Test Scenarios:

is_even(2) → True is_even(7) → False is_even(0) → True

is_even(-4) → True

is_even(9) → False

Expected Output



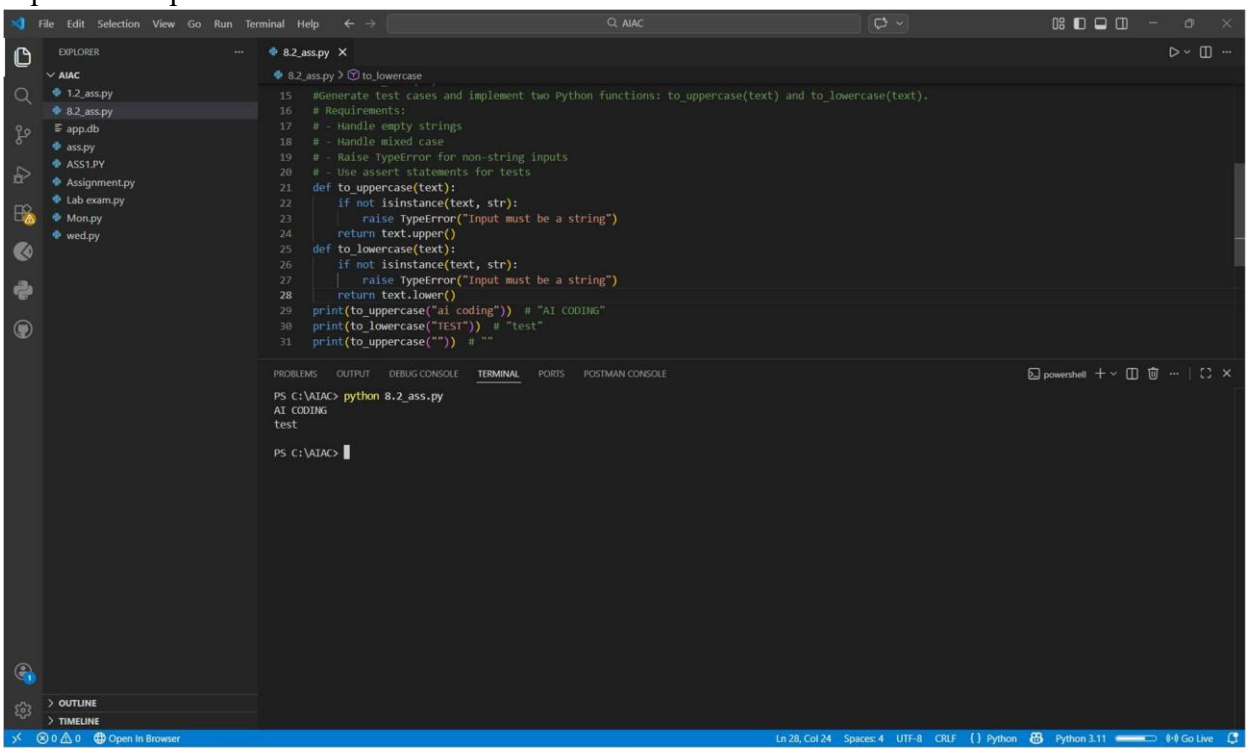
• A correctly implemented is_even() function that passes all AI-generated test cases

Task Description

Task 2 – Test-Driven Development for String Case Converter

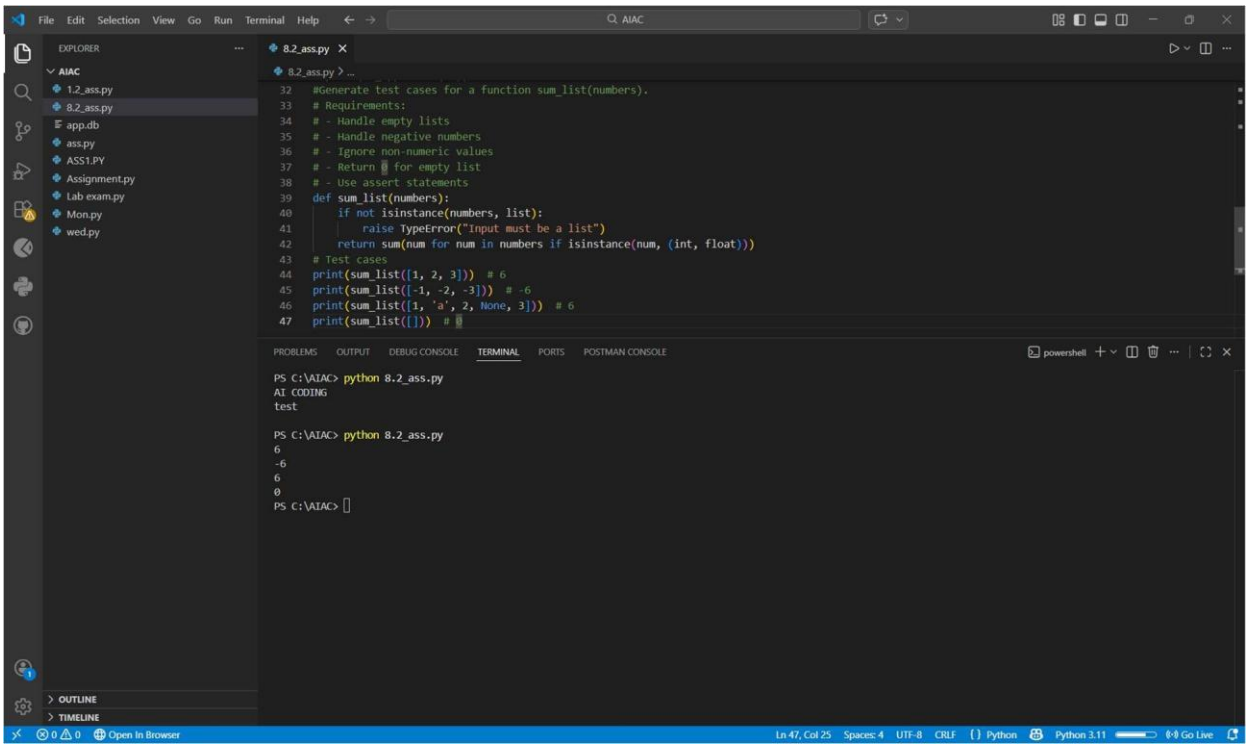
• Ask AI to generate test cases for two functions:

• to_uppercase(text)
• to_lowercase(text) Requirements:
• Handle empty strings
• Handle mixed-case input
• Handle invalid inputs such as numbers or None Example Test Scenarios: to_uppercase("ai coding") → "AI CODING" to_lowercase("TEST") → "test" to_uppercase("") → ""
to_lowercase(None) → Error or safe handling
Expected Output



• Two string conversion functions that pass all AI-generated test cases with safe input handling.

Task Description
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 Example Test Scenarios: sum_list([1, 2, 3]) → 6 sum_list([]) → 0 sum_list([-1, 5, -4]) → 0
sum_list([2, "a", 3]) → 5
Expected Output



• A robust list-sum function validated using AI-generated test cases.

Task Description

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:

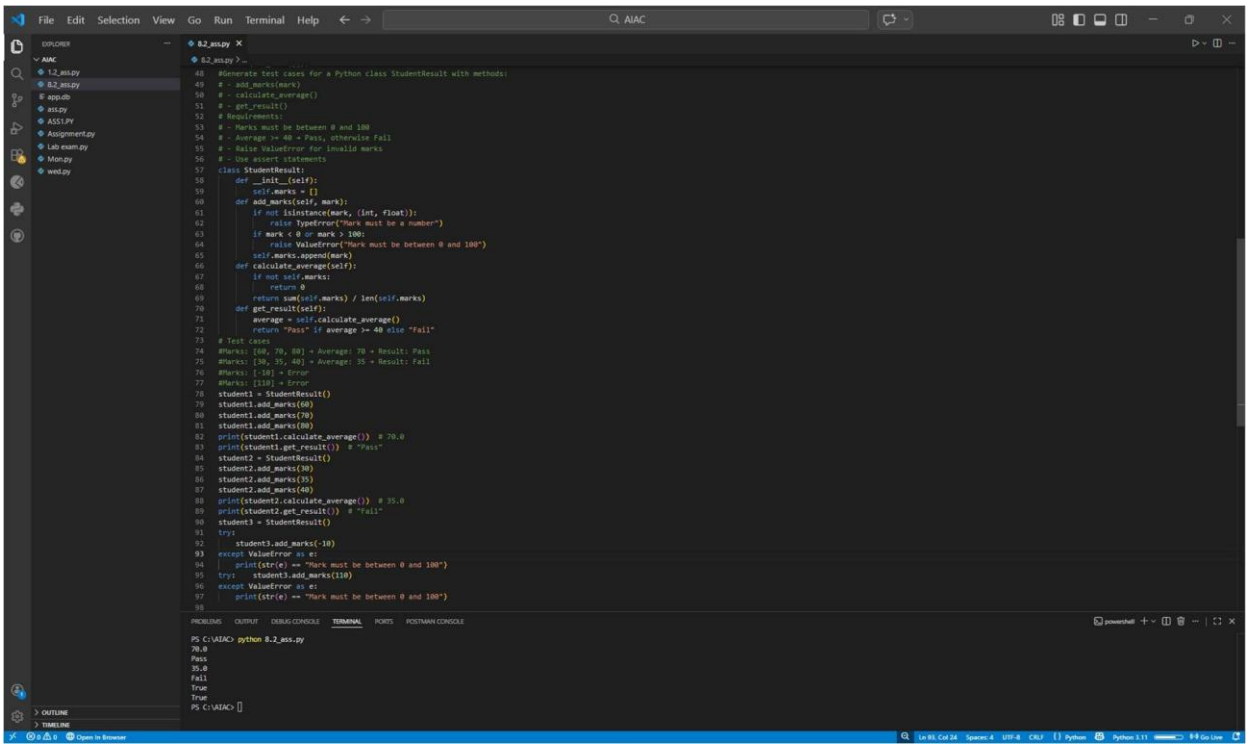
• Marks must be between 0 and 100 • Average ≥ 40 → Pass, otherwise Fail Example Test Scenarios:

Marks: [60, 70, 80] → Average: 70 → Result: Pass

Marks: [30, 35, 40] → Average: 35 → Result: Fail

Marks: [-10] → Error

Expected Output



• A fully functional StudentResult class that passes all AI-generated test

Task Description

Task 5 – Test-Driven Development for Username Validator

Requirements:

• Minimum length: 5 characters

• No spaces allowed

• Only alphanumeric characters Example Test Scenarios:

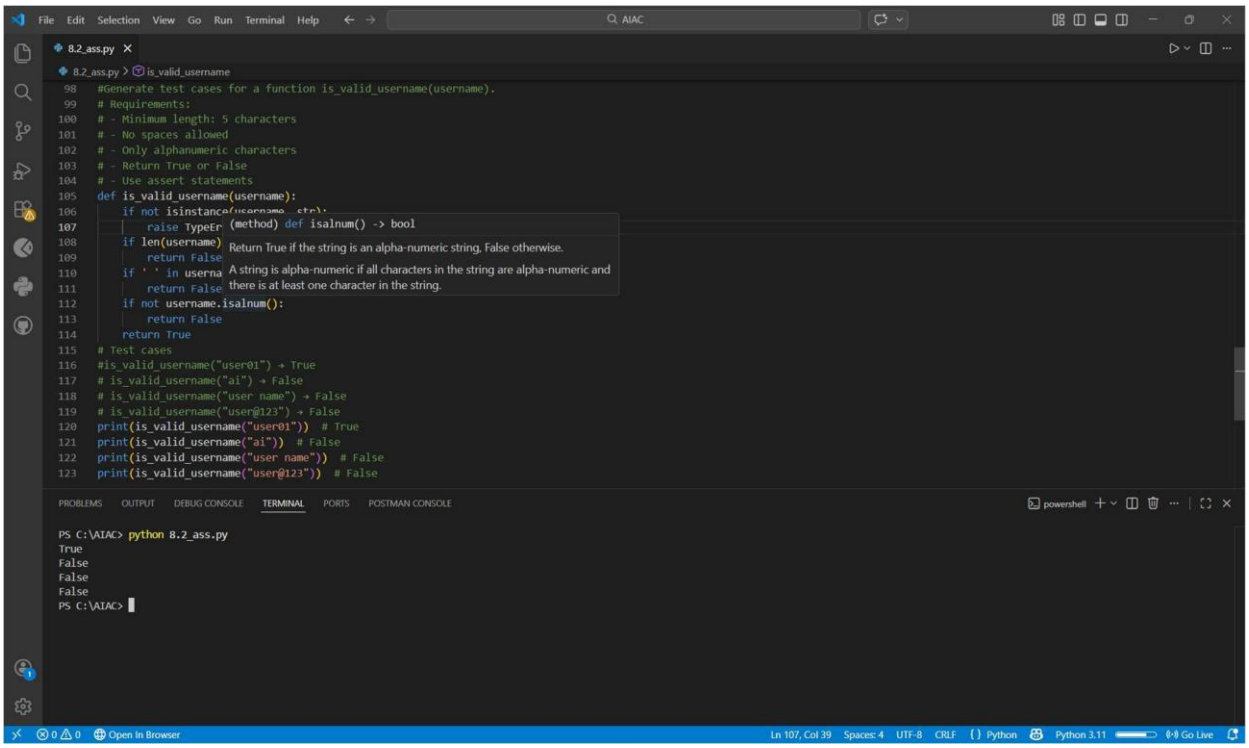
is_valid_username("user01") → True

is_valid_username("ai") → False

is_valid_username("user name") → False

is_valid_username("user@123") → False

Expected Output


File Edit Selection View Go Run Terminal Help
AIAC
8.2_ass.py
8.2_ass.py > is_valid_username
#Generate test cases for a function is_valid_username(username).
# Requirements:
# - Minimum length: 5 characters
# - No spaces allowed
# - Only alphanumeric characters
# - Return True or False
# - Use assert statements
def is_valid_username(username):
if not isinstance(username, str):
raise TypeEr
(method) def isalnum() -> bool
Return True if the string is an alpha-numeric string, False otherwise.
A string is alpha-numeric if all characters in the string are alpha-numeric and there is at least one character in the string.
if len(username)
return False
if ' ' in userna
return False
if not username.isalnum():
return False
return True
Test cases
#is_valid_username("user01") → True
is_valid_username("ai") → False
is_valid_username("user name") → False
is_valid_username("user@123") → False
print(is_valid_username("user01")) # True
print(is_valid_username("ai")) # False
print(is_valid_username("user name")) # False
print(is_valid_username("user@123")) # False
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
powershell
PS C:\AIAC> python 8.2_ass.py
True
False
False
False
PS C:\AIAC>
Open In Browser
Ln 107, Col 39 Spaces: 4 UTF-8 CRLF Python Python 3.11 Go Live