

# AI ASSISTED CODING

2303A51111

BACTH – 03

17 - 02 - 2026

## ASSIGNMENT – 08

## **LAB – 08 : Test – Driven Development with AI – Generating and Working with Test Cases.**

## **Task – 01:** Test – Driven Development for Odd/Even Number Validator.

**Prompt :** Generate unittest test cases for a Python function `is_even(n)` that checks whether a number is even. Handle zero, negative numbers, large integers, and invalid input. **Code :**

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** AIAC\_1121
- Explorer Panel (Left):**
  - OPEN EDITORS
  - AIAC\_1121\_LAB\_08.py
  - AIAC\_1121\_LAB\_08.pdf
  - 2303A51121\_AIAC\_ASSIGNMENT\_2.2.pdf
  - 2303A51121\_AIAC\_ASSIGNMENT\_2.5.pdf
  - 2303A51121\_AIAC\_ASSIGNMENT\_3.2.pdf
  - 2303A51121\_AIAC\_ASSIGNMENT\_5.5.pdf
  - 2303A51121\_AIAC\_ASSIGNMENT\_6.2.pdf
  - 2303A51121\_AIAC\_ASSIGNMENT\_7.5.pdf
  - 2303A51121\_AIAC\_ASSIGNMENT\_9.5.pdf
  - AIAC\_1121\_LAB\_08.py
  - Calculator.py
  - Factorial.py
  - Lab\_2.5.py
  - String.py
  - Task\_3.py
- Code Editor (Center):** The file AIAC\_1121\_LAB\_08.py contains Python test cases for a function is\_even(n). The code includes three test functions: test\_even\_numbers, test\_odd\_numbers, and test\_invalid\_input.
- Bottom Status Bar:** Indexing completed, No connection, In 21 Col 1, Spaces: 4, UTF-8, CRLF, Python 3.12.0, Go live

## **Output :**

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS POSTMAN CONSOLE
✖ powershell ⚠ + ⌂ ⌂ ... | ⌂ ×

● PS C:\Users\suman\OneDrive\Desktop\AIAC_1121> python AIAC_1121_LAB_08.py
● PS C:\Users\suman\OneDrive\Desktop\AIAC_1121> python AIAC_1121_LAB_08.py
...
-----
Ran 3 tests in 0.000s

OK
❖ Ran 3 tests in 0.000s

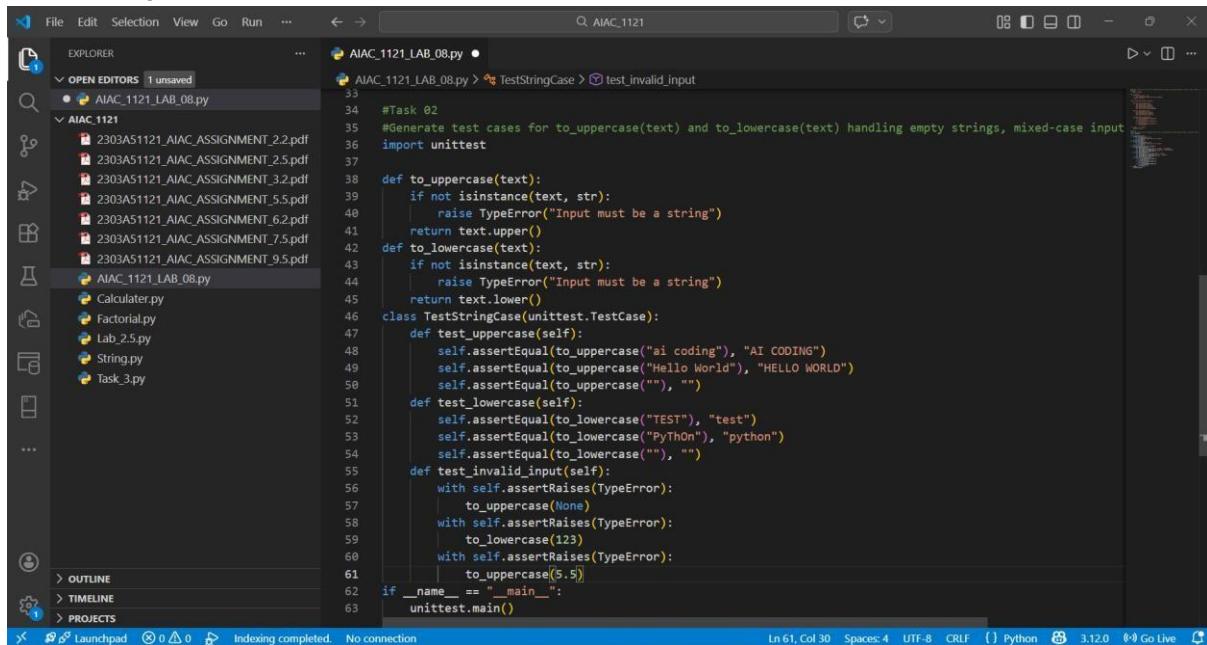
d. No connection Ln 21, Col 1 Spaces: 4 UTF-8 CRLF { } Python 🐍 3.12.0 ( ⓘ Go Live 🔍
```

## **Explanation :**

The function first validates that the input is an integer. It then checks divisibility by 2 using modulus operator. It handles zero, negative, and large integers correctly.

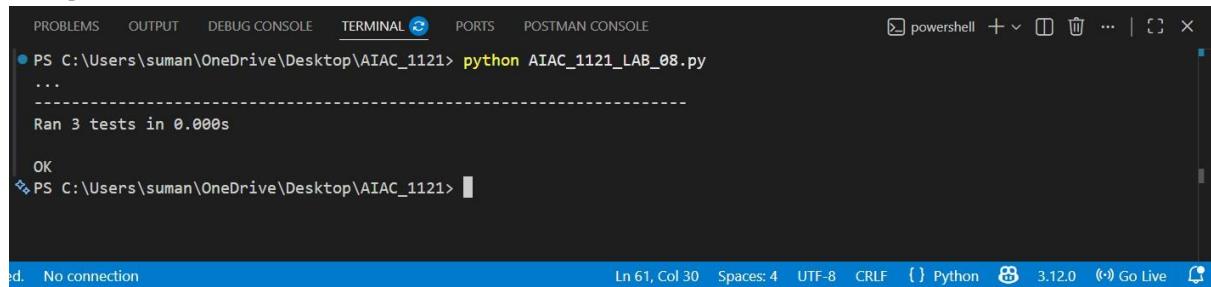
## Task – 02 : Test – Driven Development for String Case Converter.

**Prompt :** Generate test cases for `to_uppercase(text)` and `to_lowercase(text)` handling empty strings, mixed-case input, and invalid inputs. **Code :**



```
File Edit Selection View Go Run ... ← → 🔍 AIAC_1121_LAB_08.py ● AIAC_1121_LAB_08.py > 📁 TestStringCase > test_invalid_input
33 #Task 02
34 #Generate test cases for to_uppercase(text) and to_lowercase(text) handling empty strings, mixed-case input
35 import unittest
36
37 def to_uppercase(text):
38     if not isinstance(text, str):
39         raise TypeError("Input must be a string")
40     return text.upper()
41
42 def to_lowercase(text):
43     if not isinstance(text, str):
44         raise TypeError("Input must be a string")
45     return text.lower()
46
47 class TestStringCase(unittest.TestCase):
48     def test_uppercase(self):
49         self.assertEqual(to_uppercase("ai coding"), "AI CODING")
50         self.assertEqual(to_uppercase("Hello World"), "HELLO WORLD")
51         self.assertEqual(to_uppercase(""), "")
52
53     def test_lowercase(self):
54         self.assertEqual(to_lowercase("TEST"), "test")
55         self.assertEqual(to_lowercase("PyThOn"), "python")
56         self.assertEqual(to_lowercase(""), "")
57
58     def test_invalid_input(self):
59         with self.assertRaises(TypeError):
60             to_uppercase(None)
61         with self.assertRaises(TypeError):
62             to_lowercase(123)
63         with self.assertRaises(TypeError):
64             to_uppercase(5.5)
65
66 if __name__ == "__main__":
67     unittest.main()
```

## Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell + ⌂ ⌂ ⌂ ...
PS C:\Users\suman\OneDrive\Desktop\AIAC_1121> python AIAC_1121_LAB_08.py
...
-----
Ran 3 tests in 0.000s

OK
PS C:\Users\suman\OneDrive\Desktop\AIAC_1121>
```

## Explanation :

Both functions validate input type and use built-in string methods `.upper()` and `.lower()` for conversion.

## Task – 03 : Test – Driven Development for List sum Calculator.

**Prompt :** Generate test cases for `sum_list(numbers)` that handles

empty lists, negative numbers, and ignores non-numeric values.

## **Code :**

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files in the current workspace, including `AIAC_1121_ASSIGNMENT_2_2.pdf`, `AIAC_1121_ASSIGNMENT_2_5.pdf`, `AIAC_1121_ASSIGNMENT_3_2.pdf`, `AIAC_1121_ASSIGNMENT_5_5.pdf`, `AIAC_1121_ASSIGNMENT_6_2.pdf`, `AIAC_1121_ASSIGNMENT_7_5.pdf`, `AIAC_1121_ASSIGNMENT_9_5.pdf`, `AIAC_1121_LAB_08.py`, `Calculator.py`, `Factorial.py`, `Lab_2_5.py`, `String.py`, and `Task_3.py`.
- Editor:** Displays the file `AIAC_1121_LAB_08.py`. The code defines a `sum_list` function that takes a list of numbers and returns their sum, ignoring non-numeric values. It also includes a `TestSumList` class with various test methods for different input scenarios.
- Status Bar:** Shows the following information: Line 85, Column 51; Spaces: 4; UTF-8; CRLF; Python 3.12.0; Go Live.

## **Output :**

## **Explanation :**

The function iterates through the list and adds only numeric values. It safely ignores non-numeric elements and returns 0 for empty lists.

## **Task - 04 : Test Cases for Student Result Class.**

**Prompt :** Generate test cases for a StudentResult class with methods: add\_marks, calculate\_average, get\_result. Marks must be between 0 and 100.

**Code:**

The screenshot shows the PyCharm IDE interface. The left sidebar displays the 'EXPLORER' view with project files like 'AIAC\_1121\_ASSIGNMENT\_22.pdf', 'AIAC\_1121\_ASSIGNMENT\_23.pdf', 'AIAC\_1121\_ASSIGNMENT\_54.pdf', 'AIAC\_1121\_ASSIGNMENT\_62.pdf', 'AIAC\_1121\_ASSIGNMENT\_75.pdf', 'AIAC\_1121\_ASSIGNMENT\_95.pdf', 'AIAC\_1121\_LAB\_08.py', 'CalculateAvg.py', 'Factorial.py', 'Lab\_25.py', 'String.py', and 'Task\_3.py'. The top navigation bar includes 'File', 'Edit', 'Selection', 'View', 'Go', and various icons. The main area shows the code for 'AIAC\_1121\_LAB\_08.py'. The code defines a 'StudentResult' class with methods for adding marks, calculating averages, and getting results. It also includes test cases for the 'TestStudentResult' class using the 'unittest' module. The status bar at the bottom indicates 'Indexing completed. No connection'.

```
File Edit Selection View Go ... AIAC_1121_LAB_08.py AIAC_1121_LAB_08.py > TestStudentResult > test_invalid_marks

#Task 04
#Generate test cases for a StudentResult class with methods: add_marks, calculate_average, get_result. Marks must be between 0 and 100.

import unittest

class StudentResult:
    def __init__(self):
        self.marks = []
    def add_marks(self, mark):
        if not isinstance(mark, (int, float)) or mark < 0 or mark > 100:
            raise ValueError("Marks must be between 0 and 100")
        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):
        return "Pass" if self.calculate_average() >= 40 else "Fail"

class TestStudentResult(unittest.TestCase):
    def test_pass_case(self):
        student = StudentResult()
        student.add_marks(50)
        student.add_marks(70)
        student.add_marks(90)
        student.add_marks(80)
        self.assertEqual(student.calculate_average(), 70)
        self.assertEqual(student.get_result(), "Pass")
    def test_fail_case(self):
        student = StudentResult()
        student.add_marks(50)
        student.add_marks(55)
        student.add_marks(55)
        student.add_marks(45)
        self.assertEqual(student.calculate_average(), 55)
        self.assertEqual(student.get_result(), "Fail")
    def test_invalid_marks(self):
        student = StudentResult()
        with self.assertRaises(ValueError):
            student.add_marks(-10)
        with self.assertRaises(ValueError):
            student.add_marks(100)
        with self.assertRaises(ValueError):
            student.add_marks('100')
        with self.assertRaises(ValueError):
            student.add_marks(["100"])

if __name__ == "__main__":
    unittest.main()
```

## Output:

## **Explanation :**

The class validates marks before storing them. It calculates average dynamically and determines result based on 40% threshold.

## **Task – 05 : Test – Driven Development for username Validator.**

**Prompt:** Generate test cases for `is_valid_username(username)` with minimum 5 characters, no spaces, and only alphanumeric characters.

## Code :

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files in the project. Files include AIAC\_1121\_LAB\_08.py, Calculator.py, Factorial.py, Lab\_2.5.py, String.py, Task\_3.py, and several PDF files (2303A51121\_AIAC\_ASSIGNMENT\_2.2.pdf, 2303A51121\_AIAC\_ASSIGNMENT\_2.5.pdf, 2303A51121\_AIAC\_ASSIGNMENT\_3.2.pdf, 2303A51121\_AIAC\_ASSIGNMENT\_5.5.pdf, 2303A51121\_AIAC\_ASSIGNMENT\_6.2.pdf, 2303A51121\_AIAC\_ASSIGNMENT\_7.5.pdf, 2303A51121\_AIAC\_ASSIGNMENT\_9.5.pdf).
- Code Editor:** Displays the content of AIAC\_1121\_LAB\_08.py. The code defines a function `is_valid_username` and a test class `TestUsernameValidator`. The function checks if the input is a string, has at least 5 characters, contains no spaces, and is alphanumeric. The test class contains methods for testing valid, short, and special character usernames, as well as invalid length and type cases.
- Status Bar:** Shows "Indexing completed.", "No connection", "Ln 167, Col 51", "Spaces: 4", "UTF-8", "CRLF", "Python 3.12.0", and "Go Live".

## Output :

The screenshot shows a terminal window with the following output:

```
● PS C:\Users\suman\OneDrive\Desktop\AIAC_1121> python AIAC_1121_LAB_08.py
.....
Ran 5 tests in 0.000s
OK
PS C:\Users\suman\OneDrive\Desktop\AIAC_1121>
```

The terminal shows the command `python AIAC_1121_LAB_08.py` being run, followed by the results of the test suite. It indicates 5 tests ran in 0.000 seconds and everything was successful ("OK").

Status bar: "d. No connection", "Ln 169, Col 20", "Spaces: 4", "UTF-8", "CRLF", "Python 3.12.0", "Go Live".

## Explanation :

The function checks length, space restriction, and alphanumeric condition using built-in string validation methods.

**THANK YOU!!**