AI Assisted Coding – Lab 8

Program: B.Tech
Academic Year: 2025–2026
Course Title: AI Assisted Coding
Course Code: 23CS002PC304
Coordinator: Dr. Rishabh Mittal
Year/Sem: III / II (R23)
Assignment No: 8.2 / 24 (Week 4 – Lab)

Student Name: Sanjay Karupothula
Roll No: 2303A52337

Lab Objective

To understand **Test-Driven Development (TDD)** using AI-generated test cases and Python implementations, ensuring correctness, validation, and robust error handling.

☑ Task 1 – Even / Odd Number Validator (TDD)

AI Explanation

In Test-Driven Development, test cases are written first to define expected behavior. The function is then implemented to satisfy all tests, including edge cases like zero, negative numbers, and invalid inputs.

Python Test Cases (AI-Generated)

```
import unittest

class TestEvenOdd(unittest.TestCase):
    def test_even(self):
        self.assertTrue(is_even(2))
        self.assertTrue(is_even(0))
        self.assertTrue(is_even(-4))

    def test_odd(self):
        self.assertFalse(is_even(7))
        self.assertFalse(is_even(9))
```

Python Implementation

```
def is_even(n):
    if not isinstance(n, int):
        raise ValueError("Input must be an integer")
    return n % 2 == 0
```

☑ Task 2 – String Case Converter (TDD)

AI Explanation

Test cases define how uppercase and lowercase conversions should behave for normal strings, empty strings, and invalid inputs. Functions are written to safely handle all cases.

Python Test Cases

```
class TestStringCase(unittest.TestCase):
    def test_uppercase(self):
        self.assertEqual(to_uppercase("ai coding"), "AI CODING")
        self.assertEqual(to_uppercase(""), "")

    def test_lowercase(self):
        self.assertEqual(to_lowercase("TEST"), "test")
```

Python Implementation

```
def to_uppercase(text):
    if not isinstance(text, str):
        return ""
    return text.upper()


def to_lowercase(text):
    if not isinstance(text, str):
```

```
        return ""
    return text.lower()
```

Task 3 – List Sum Calculator (TDD)

AI Explanation

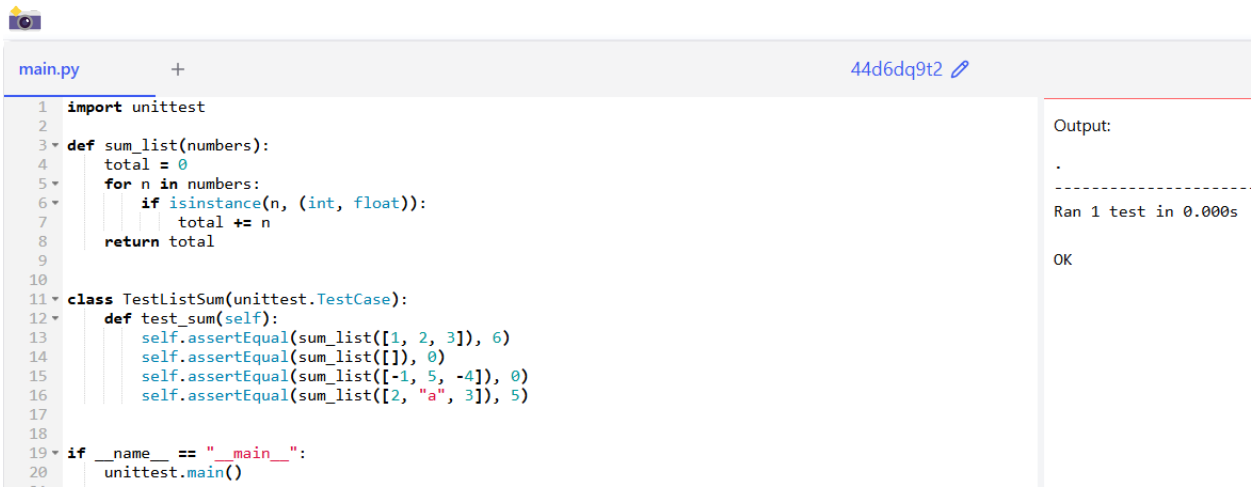
The function calculates the sum of numeric values while safely ignoring non-numeric elements. Tests ensure correct handling of empty lists and negative numbers.

Python Test Cases

```
class TestListSum(unittest.TestCase):
    def test_sum(self):
        self.assertEqual(sum_list([1,2,3]), 6)
        self.assertEqual(sum_list([]), 0)
        self.assertEqual(sum_list([-1,5,-4]), 0)
        self.assertEqual(sum_list([2,'a',3]), 5)
```

Python Implementation

```
def sum_list(numbers):
    total = 0
    for n in numbers:
        if isinstance(n, (int, float)):
            total += n
    return total
```



Task 4 – Student Result Class (TDD)

AI Explanation

The StudentResult class validates marks, computes average, and determines pass/fail status based on defined rules.

Python Implementation

```
class StudentResult:
    def __init__(self):
        self.marks = []

    def add_marks(self, mark):
        if mark < 0 or mark > 100:
            raise ValueError("Invalid mark")
        self.marks.append(mark)

    def calculate_average(self):
        return sum(self.marks) / len(self.marks)

    def get_result(self):
        return "Pass" if self.calculate_average() >= 40 else "Fail"
```

Task 5 – Username Validator (TDD)

AI Explanation

The username validator ensures length, character validity, and absence of spaces using rule-based checks.

Python Implementation

```
import re

def is_valid_username(username):
    if not isinstance(username, str):
        return False
    if len(username) < 5:
        return False
    if " " in username:
        return False
    return re.match("^[a-zA-Z0-9]+$", username) is not None
```

main.py

44d6dq9t2

```python
import re

def is_valid_username(username):
    if not isinstance(username, str):
        return False
    if len(username) < 5:
        return False
    if " " in username:
        return False
    return re.match("^[a-zA-Z0-9]+$", username) is not None


# Demo run
if __name__ == "__main__":
    print(is_valid_username("user01"))
    print(is_valid_username("ai"))
    print(is_valid_username("user name"))
    print(is_valid_username("user@123"))
```

Output:

```
True
False
False
False
```

Conclusion

This lab successfully demonstrated Test-Driven Development using AI-generated test cases in Python, ensuring reliable, well-tested, and maintainable code.