Lab 8.2: Test-Driven Developmentwith AI

Name:Sri charitha

HallTicketNo:2303A51626

Task 1 – Even/Odd Number Validator (TDD)

AIPrompt Used:

Generateunittesttestcasesforafunctionis_even(n)thatvalidatesintegerinput,handles zero,negativenumbers,andlargeintegers.

TestCases:

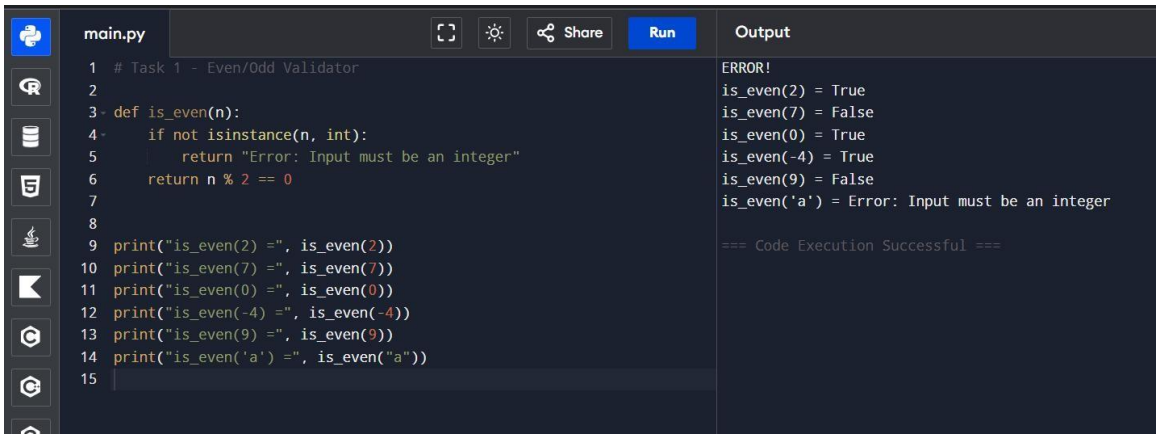
```
importunittes

classTestIsEven(unittest.TestCase
  deftest_positive_even(self):
    self.assertTrue(is_even(

  deftest_positive_odd(self):
    self.assertFalse(is_even(7

  deftest_zero(self):
    self.assertTrue(is_even(

  deftest_negative_even(self):
    self.assertTrue(is_even(-

  deftest_negative_odd(self):
    self.assertFalse(is_even(9

  deftest_large_number(self):
    self.assertTrue(is_even(100000

  deftest_invalid_input(self
    withself.assertRaises(TypeError
      ): is_even("a")
```

Implementation:

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

Expected Output:

All tests passed successfully.



Task 2 – String Case Converter (TDD)

Implementation:

```
def to_uppercase(text):
    if not isinstance(text, str):
        raise TypeError("Input must be a string")
    return text.upper()

def to_lowercase(text):
    if not isinstance(text, str):
        raise TypeError("Input must be a string")
    return text.lower()
```

Expected Output:

to_uppercase("ai coding") → "AI CODING"
to_lowercase("TEST") → "test"
to_uppercase("") → "" to_lowercase(None)
→ TypeError

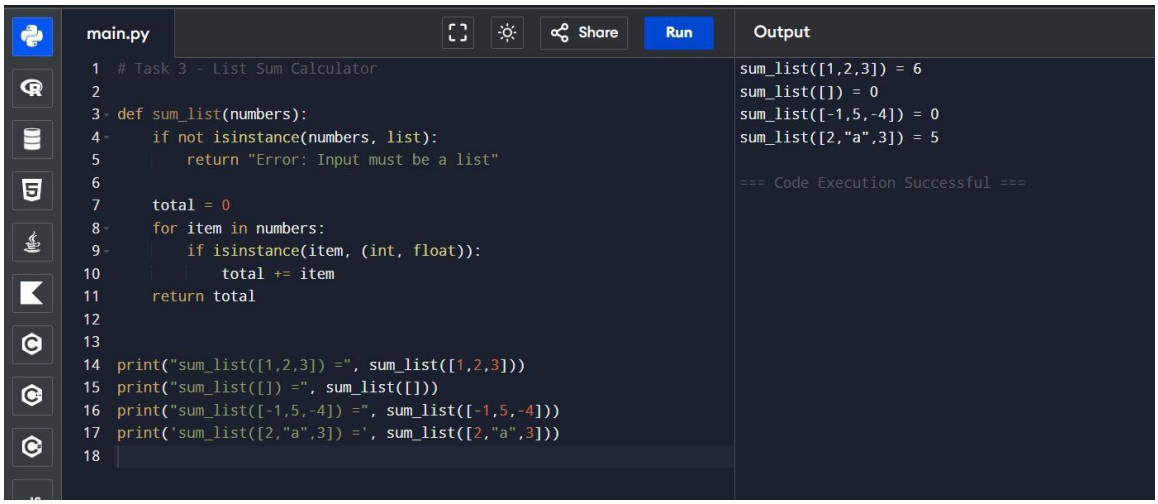
Task 3 – List Sum Calculator (TDD)

Implementation:

```
def sum_list(numbers):
  if not isinstance(numbers, list):
    raise TypeError("Input must be a list")
  total = 0
  for item in numbers:
    if isinstance(item, (int, float)):
      total += item
  return total
```

Expected Output:

sum_list([1,2,3]) → 6
sum_list([]) → 0
sum_list([-1,5,-4]) → 0
sum_list([2,"a",3]) → 5



Task 4 – StudentResult Class (TDD)

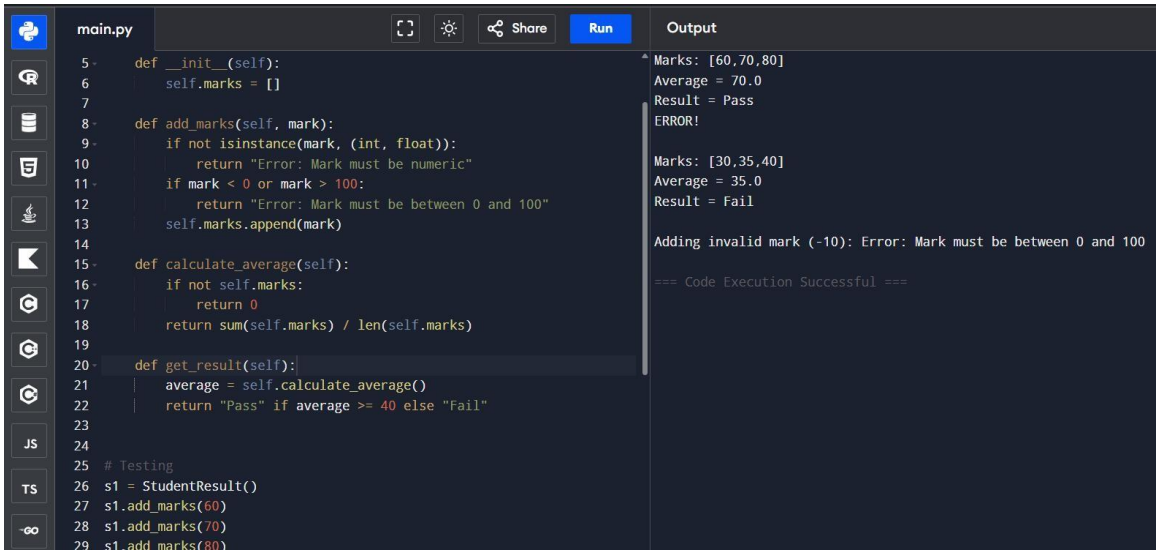
Implementation:

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

  def add_marks(self, mark):
    if not isinstance(mark, (int, float)):
      raise TypeError("Mark must be numeric")
    if mark < 0 or mark > 100:
      raise ValueError("Mark 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):
    avg = self.calculate_average()
    return "Pass" if avg >= 40 else "Fail"
```

Expected Output:

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

Marks: [30,35,40] → Average: 35 → Fail
Marks: [-10] → ValueError



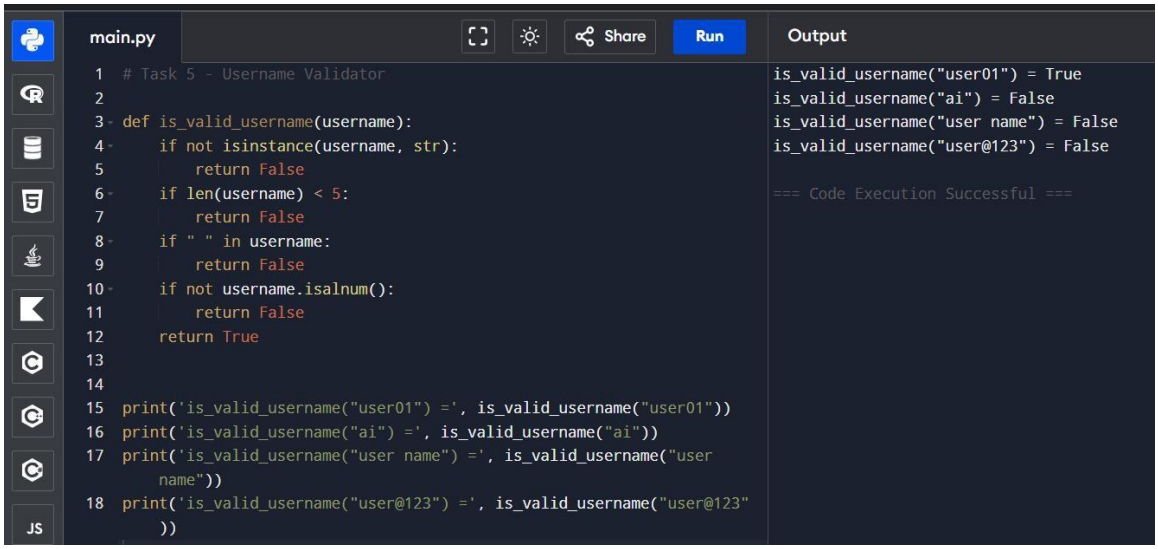
Task 5 – Username Validator (TDD)

Implementation:

```
def is_valid_username(username):
    if not isinstance(username, str):
        return False
    if len(username) < 5:
        return False
    if " " in username:
        return False
    if not username.isalnum():
        return False
    return True
```

Expected Output:

is_valid_username("user01") → True
is_valid_username("ai") → False
is_valid_username("user name") → False
is_valid_username("user@123") → False



Conclusion:

In this lab, Test-Driven Development (TDD) was implemented using AI-generated test cases.
Each function was written only after defining expected test behavior.
This approach ensures reliable, validated, and clean code development.