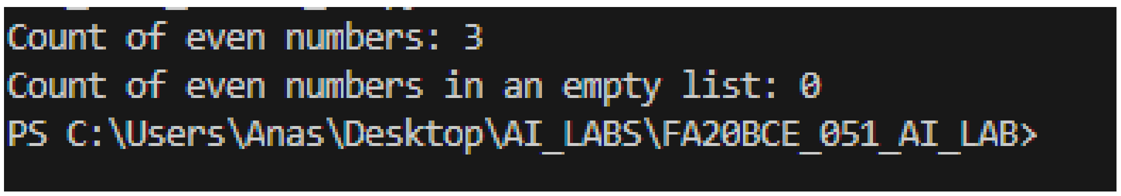
Lab Task 1

```
 def count_even_numbers(number_list):
    # Check if the list is empty
    if len(number_list) == 0:
        return 0

    # Initialize a counter variable
    even_count = 0

    # Iterate through each number in the list
    for number in number_list:
        # Check if the number is even
        if number % 2 == 0:
            even_count += 1

    # Return the count of even numbers
    return even_count

# Example usage:
my_list = [2,2,2,3,5]
result = count_even_numbers(my_list)
print("Count of even numbers:", result)

empty_list = []
result_empty = count_even_numbers(empty_list)
print("Count of even numbers in an empty list:", result_empty)
```

Output :



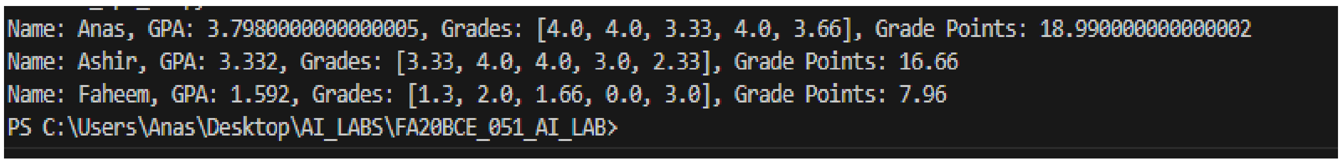
```
Count of even numbers: 3
Count of even numbers in an empty list: 0
PS C:\Users\Anas\Desktop\AI_LABS\FA20BCE_051_AI_LAB>
```

Lab Task 2

```
def calculate_gpa(students):

    def calculate_grade_points(marks):
        if marks >= 85:
            return 4.00
        elif marks >= 80:
            return 3.66
        elif marks >= 75:
            return 3.33
        elif marks >= 71:
            return 3.00
        elif marks >= 68:
            return 2.66
        elif marks >= 64:
            return 2.33
        elif marks >= 61:
            return 2.00
        elif marks >= 58:
            return 1.66
        elif marks >= 54:
            return 1.30
        elif marks >= 50:
            return 1.00
        else:
            return 0.00

    result = []
    for student in students:
        name = student['name']
        marks = student['marks']
        total_grade_points = sum([calculate_grade_points(mark) for mark in marks])
        gpa = total_grade_points / len(marks)
        grades = [calculate_grade_points(mark) for mark in marks]

        result.append({
            'name': name,
            'grades': grades,
            'grade_points': total_grade_points,
            'gpa': gpa
        })
```

```python
    return result

# Example usage:
students = [
    {'name': 'Anas', 'marks': [90, 88, 76, 95, 82]},
    {'name': 'Ashir', 'marks': [78, 85, 92, 71, 65]},
    {'name': 'Faheem', 'marks': [55, 62, 58, 49, 73]}
]

gpa_results = calculate_gpa(students)
for result in gpa_results:
    print(f"Name: {result['name']}, GPA: {result['gpa']}, Grades:
{result['grades']}, Grade Points: {result['grade_points']}")
```

Output :



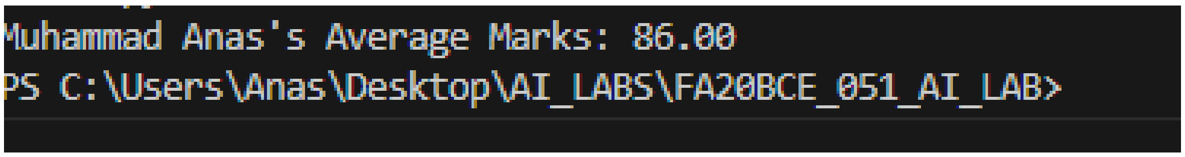
```
Name: Anas, GPA: 3.7980000000000005, Grades: [4.0, 4.0, 3.33, 4.0, 3.66], Grade Points: 18.990000000000002
Name: Ashir, GPA: 3.332, Grades: [3.33, 4.0, 4.0, 3.0, 2.33], Grade Points: 16.66
Name: Faheem, GPA: 1.592, Grades: [1.3, 2.0, 1.66, 0.0, 3.0], Grade Points: 7.96
PS C:\Users\Anas\Desktop\AI_LABS\FA20BCE_051_AI_LAB>
```

Lab Task 3

Student

```python
class Student:

    def __init__(self, name, roll_number):
        self.name = name
        self.roll_number = roll_number
        self.marks = []

    def add_marks(self, subject, mark):
        self.marks.append((subject, mark))

    def calculate_average(self):
        if not self.marks:
            return 0.0
        total_marks = sum(mark for _, mark in self.marks)
        average = total_marks / len(self.marks)
        return average

# Create an instance of the Student class
student1 = Student("Muhammad Anas", "051")

# Add some marks
student1.add_marks("Math", 90)
student1.add_marks("Science", 90)
student1.add_marks("History", 78)

# Calculate and print the average marks
average_marks = student1.calculate_average()
print(f"{student1.name}'s Average Marks: {average_marks:.2f}")
```

Output:



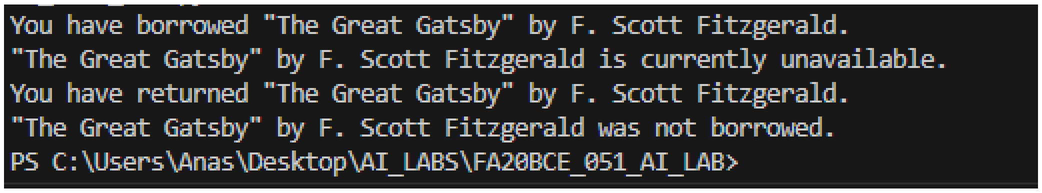
```
Muhammad Anas's Average Marks: 86.00
PS C:\Users\Anas\Desktop\AI_LABS\FA20BCE_051_AI_LAB>
```

Post-Lab:

Book:

```python
 class Book:

    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.available = True

    def borrow(self):
        if self.available:
            self.available = False
            return f'You have borrowed "{self.title}" by {self.author}.'
        else:
            return f'"{self.title}" by {self.author} is currently
unavailable.'

    def return_book(self):
        if not self.available:
            self.available = True
            return f'You have returned "{self.title}" by {self.author}.'
        else:
            return f'"{self.title}" by {self.author} was not borrowed.'

# Example usage:
book1 = Book('The Great Gatsby', 'F. Scott Fitzgerald')
print(book1.borrow())
print(book1.borrow())
print(book1.return_book())
print(book1.return_book())
```

Output:



```
You have borrowed "The Great Gatsby" by F. Scott Fitzgerald.
"The Great Gatsby" by F. Scott Fitzgerald is currently unavailable.
You have returned "The Great Gatsby" by F. Scott Fitzgerald.
"The Great Gatsby" by F. Scott Fitzgerald was not borrowed.
PS C:\Users\Anas\Desktop\AI_LABS\FA20BCE_051_AI_LAB>
```