

## In-Lab

### Task 1

Write the following code in Spyder IDE

```
def count_even_number(number_list):  
    even_count = 0  
  
    for num in number_list:  
  
        if num % 2 == 0:  
  
            even_count += 1;  
    return even_count  
  
numbers1 = [1,2,3,4,5,6,7,8,9]  
numbers2 = [42,421,52,523,23,23]  
  
print(count_even_number(numbers1))  
print(count_even_number(numbers2))
```

After executing the program, you should see an output similar to the following image.



```
4  
2
```

### Task 2

Write the following code in Spyder IDE

```
def calculate_gpa(students):  
    grading_scale = {  
        (85, 100): ('A', 4.00),  
        (80, 84): ('A-', 3.66),  
        (75, 79): ('B+', 3.33),  
        (71, 74): ('B', 3.00),  
        (68, 70): ('B-', 2.66),  
        (64, 67): ('C+', 2.33),  
        (61, 63): ('C', 2.00),  
        (58, 60): ('C-', 1.66),  
        (54, 57): ('D+', 1.30),  
        (50, 53): ('D', 1.00),
```

```
(0, 49): ('F', 0.00),
}

result = []

for student in students:
    name = student['name']
    marks = student['marks']
    total_grade_points = 0
    grades = []

    for mark in marks:
        for score_range, (grade, grade_point) in grading_scale.items():
            if score_range[0] <= mark <= score_range[1]:
                total_grade_points += grade_point
                grades.append(grade)
                break

    gpa = total_grade_points / len(marks)

    student_info = {
        'name': name,
        'grades': grades,
        'grade_points': total_grade_points,
        'gpa': round(gpa, 2)
    }

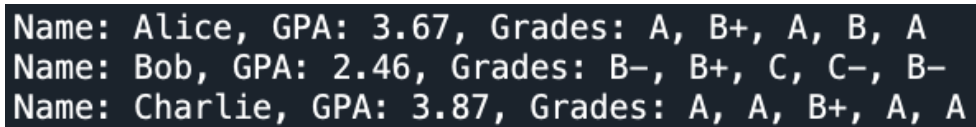
    result.append(student_info)

return result

# Example usage:
students = [
    {'name': 'Alice', 'marks': [92, 78, 85, 73, 88]},
    {'name': 'Bob', 'marks': [68, 75, 62, 58, 70]},
    {'name': 'Charlie', 'marks': [95, 88, 79, 87, 91]},
]

gpa_results = calculate_gpa(students)
for student in gpa_results:
    print(f"Name: {student['name']}, GPA: {student['gpa']}, Grades: {'',
        '.join(student['grades'])}")
```

After executing the program, you should see an output similar to the following image.



```
Name: Alice, GPA: 3.67, Grades: A, B+, A, B, A
Name: Bob, GPA: 2.46, Grades: B-, B+, C, C-, B-
Name: Charlie, GPA: 3.87, Grades: A, A, B+, A, A
```

### Task 3

Write the following code in Spyder IDE

```
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 subject, mark in self.marks)
        return total_marks / len(self.marks)

# Create an instance of the Student class
student1 = Student("Omar Akbar Have ", "13232")

# Add marks for different subjects
student1.add_marks("Machine Learning", 70)
student1.add_marks("Digital System Design", 85)
student1.add_marks("Image Processing", 50)
student1.add_marks("IOT", 72)

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

After executing the program, you should see an output similar to the following image.

## Post-Lab

### Task 1

Write the following code in Spyder IDE

```
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
            print(f"You have borrowed '{self.title}' by {self.author}.")
        else:
            print(f"'{self.title}' is currently not available.")

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

    def check_availability(self):
        if self.available:
            print(f"'{self.title}' by {self.author} is available.")
        else:
            print(f"'{self.title}' by {self.author} is currently not available.")

# Example usage:
book1 = Book("The ABC", "Ali")
book2 = Book("Hellow World", "Zain")

book1.check_availability() # Check availability of book1
book1.borrow()            # Borrow book1
book1.borrow()            # Try to borrow book1 again (already borrowed)
book1.return_book()       # Return book1
book1.return_book()       # Try to return book1 again (already returned)

book2.check_availability() # Check availability of book2
book2.borrow()            # Borrow book2
book2.return_book()       # Return book2
book2.check_availability() # Check availability of book2 again
```

```
'The ABC' by Ali is available.  
You have borrowed 'The ABC' by Ali.  
'The ABC' is currently not available.  
You have returned 'The ABC' by Ali.  
'The ABC' is already available.  
'Hellow World' by Zain is available.  
You have borrowed 'Hellow World' by Zain.  
You have returned 'Hellow World' by Zain.  
'Hellow World' by Zain is available.
```