AI Assistant Coding

Assignment 10.1

Name :V. Hansika **HT. No :** 2303A52409 **Batch:** 32

Task Description #1 – Syntax and Logic Errors

Task: Use AI to identify and fix syntax and logic errors in a faulty Python script.

Sample Input Code: # Calculate average score of a student

```
def calc_average(marks):
        total = 0
        for m in marks:
                total += m
        average = total / len(marks)
        return avrage # Typo here
marks = [85, 90, 78, 92]
print("Average Score is ", calc_average(marks)
```

Expected Output:

• Corrected and runnable Python code with explanations of the fixes.

Code:

```
# Calculate average score of a student
def calc_average(marks):
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return average # Typo fixed
marks = [85, 90, 78, 92] #output shpuld be 86.25
print("Average Score is ", calc_average(marks))
```

Output:

```
Average Score is  86.25
```

Task Description #2 – PEP 8 Compliance

Task: Use AI to refactor Python code to follow PEP 8 style guidelines.

Sample Input Code:

def area_of_rect(L,B) : return L*B

print(area_of_rect(10,20))

Expected Output:

• Well-formatted PEP 8-compliant Python code.

Code:

```
#refactor the above code with proper function definition and docstring in
google style comments.
def area_of_rectangle(length: float, breadth: float) -> float:
    """
    Calculate the area of a rectangle.

    Args:
        length (float): The length of the rectangle.
        breadth (float): The breadth of the rectangle.

    Returns:
        float: The area of the rectangle.
    raises:
        ValueError: If length or breadth is negative.
        TypeError: If length or breadth is not a number.

    """

    if not isinstance(length, (int, float)) or not isinstance(breadth, (int,
float)):
        raise TypeError("Length and breadth must be numbers.")

    if length < 0 or breadth < 0:
        raise ValueError("Length and breadth must be non-negative.")
    return length * breadth
# Example usage
try:
    print(area_of_rectangle(-10, 20))
except ValueError as e:
    print(e)
try:
    print(area_of_rectangle("10", 20))
except TypeError as e:
    print(e)
```

Task Description #3 – Readability Enhancement

Task: Use AI to make code more readable without changing its logic.

Sample Input Code:

```
def c(x,y):
    return x*y/100
a=200
b=15
print(c(a,b))
```

Expected Output:

• Python code with descriptive variable names, inline comments, and clear formatting.

Code:

```
#add discriptive names to the variables and function with clear formatting and
inline comments
def calculate_percentage(value, percentage):
    # This function calculates the percentage of a given value
    return value * percentage / 100
# Define the value and percentage to be calculated
total_amount = 200
percentage_to_calculate = 15
# Calculate the percentage and print the result
result = calculate_percentage(total_amount, percentage_to_calculate)
print(result)
```

Task Description #4 – Refactoring for Maintainability

Task: Use AI to break repetitive or long code into reusable functions.

Sample Input Code:

```
students = ["Alice", "Bob", "Charlie"]
print("Welcome", students[0])
print("Welcome", students[1])
print("Welcome", students[2])
```

Expected Output:

• Modular code with reusable functions.

Code:

```
#make the code reusable with a function and loop through the list of students
def welcome_students(student_list):
```

```
    # This function takes a list of students and prints a welcome message for
each student
    for student in student_list:
        print("Welcome", student)
# Define the list of students
students = ["Alice", "Bob", "Charlie"]
# Call the function to welcome each student
welcome_students(students)
```

Task Description #5 – Performance Optimization

Task: Use AI to make the code run faster.

Sample Input Code: # Find squares of numbers

```
nums = [i for i in range(1,1000000)]
start_time = time.time()
squares = []
for n in nums:
    squares.append(n**2)
print(len(squares))
end_time = time.time()
print(f"Time taken: {end_time - start_time} seconds")
```

Expected Output:

- Optimized code using list comprehensions or vectorized operations.

Code:

```
# recator the below code to optimzied code to reduce the time complexity with
list comprehension
time.sleep(1)  # Simulating some delay
nums = [i for i in range(1,1000000)]
start_time = time.time()
squares = [n**2 for n in nums]
print(len(squares))
end_time = time.time()
print(f"Time taken: {end_time - start_time} seconds")

#more optimize code
time.sleep(1)  # Simulating some delay
start_time = time.time()
print(len([n**2 for n in range(1,1000000)]))
end_time = time.time()
print(f"Time taken: {end_time - start_time} seconds")
```

Task Description #6 – Complexity Reduction

Task: Use AI to simplify overly complex logic.

Sample Input Code:

```
def grade(score):
    if score >= 90:
        return "A"
    else:
        if score >= 80:
            return "B"
        else:
            if score > 70:
                return "C"
            else:
                if score > 60:
                    return "D"
                else:
                    return "F"
```

Expected Output:

• Cleaner logic using elif or dictionary mapping**.**

Code:

```
# make the above code with reducing complexity and improving readability using
elif statements or dictionary mapping
def grade(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score > 70:
        return "C"
    elif score > 60:
        return "D"
    else:
        return "F"
score = 85
strt_time = time.time()
print(grade(score))
end_time = time.time()
print("Execution time:", end_time - strt_time, "seconds")

# Alternatively, using dictionary mapping
def grade(score):
    grade_mapping = {
        (90, 100): "A",
```

```
        (80, 89): "B",
        (70, 79): "C",
        (60, 69): "D",
        (0, 59): "F"
    }
    for range_tuple, grade_letter in grade_mapping.items():
        if range_tuple[0] <= score <= range_tuple[1]:
            return grade_letter
    return "Invalid score"
# Example usage
score = 85
strt_time = time.time()

print(grade(score))
end_time = time.time()
print("Execution time:", end_time - strt_time, "seconds")
```