Lab Assignment 10.1

HT.No: 2303A51109
Name: M. Hasini
Batch-02

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.

>>> Fixed Code :

week 10 > average.py > ...

```
13  def calc_average(marks):
14      total = 0
15      for m in marks:
16          total += m
17      average = total / len(marks)
18      return average   # Fixed typo
19
20  marks = [85, 90, 78, 92]
21  print("Average Score is ", calc_average(marks))
```

>>> Output :

```
PS C:\Users\User\OneDrive\Desktop\AIAC> &
IAC/week 10/average.py"
Average Score is  86.25
PS C:\Users\User\OneDrive\Desktop\AIAC>
```

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.

>>> Fixed code :

week 10 > aorectangle.py > ...

```python
def area_of_rectangle(length: float, breadth: float) -> float:
    """
    This function calculates the area of a rectangle given its length and breadth.
    Parameters:
    length (float): The length of the rectangle.
    breadth (float): The breadth of the rectangle.
    Returns:
    float: The area of the rectangle.
    """
    return length * breadth
print(area_of_rectangle(10, 20))

```

>>> Output :

```
PS C:\Users\User\OneDrive\Desktop\AIAC>
IAC/week 10/aorectangle.py"
200
200
PS C:\Users\User\OneDrive\Desktop\AIAC>
```

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.

>>> Fixed Code :

```
week 10 > div.py > ...
  7  def calculate_percentage(part: float, whole: float) -> float:
  8      """
  9      This function calculates the percentage of a part relative to a whole.
 10      Parameters:
 11      part (float): The part value for which the percentage is to be calculated.
 12      whole (float): The whole value against which the percentage is calculated.
 13      Returns:
 14      float: The percentage value of the part relative to the whole.
 15      """
 16      return (part * 100) / whole
 17  print(calculate_percentage(15, 200))
```

>>> Output :

```
PS C:\Users\User\OneDrive\Desktop\AIAC> &
IAC/week 10/div.py"
7.5
PS C:\Users\User\OneDrive\Desktop\AIAC>
```

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.

>>> Fixed Code :

week 10 > students.py > ...

```python
def welcome_student(student_name: str) -> None:
    """
    This function takes a student's name as input and prints a welcome message.
    Parameters:
    student_name (str): The name of the student to welcome.
    Returns:
    None
    """
    print("Welcome", student_name)
students = ["Alice", "Bob", "Charlie"]
for student in students:
    welcome_student(student)
```

>>> Output :

```
IAC/week 10/students.py"
Welcome Alice
Welcome Bob
Welcome Charlie
```

Task Description #5 – Performance Optimization

Task: Use AI to make the code run faster.

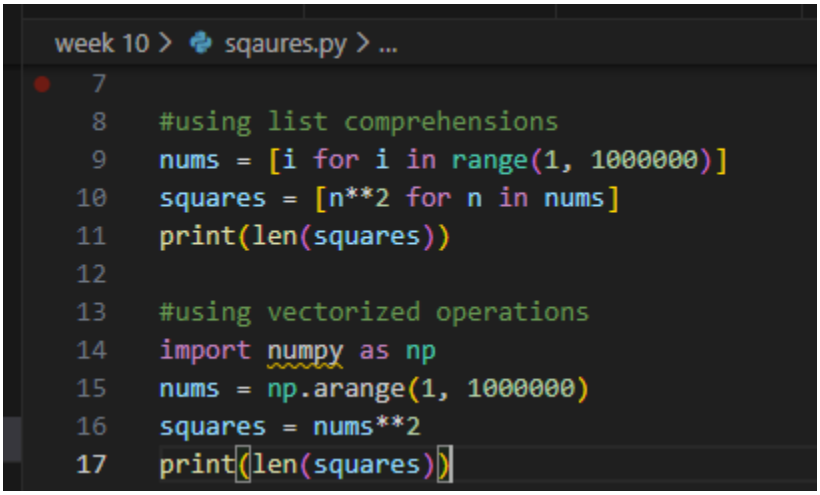
Sample Input Code:

```
# Find squares of numbers
        nums = [i for i in range(1,1000000)]
        squares = []
        for n in nums:
            squares.append(n**2)
        print(len(squares))
```

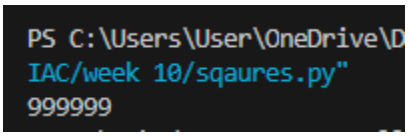
Expected Output:

- Optimized code using list comprehensions or vectorized operations.

>>> Fixed Code :



>>> Output :



Task Description #6 – Complexity Reduction

Task: Use AI to simplify overly complex logic.

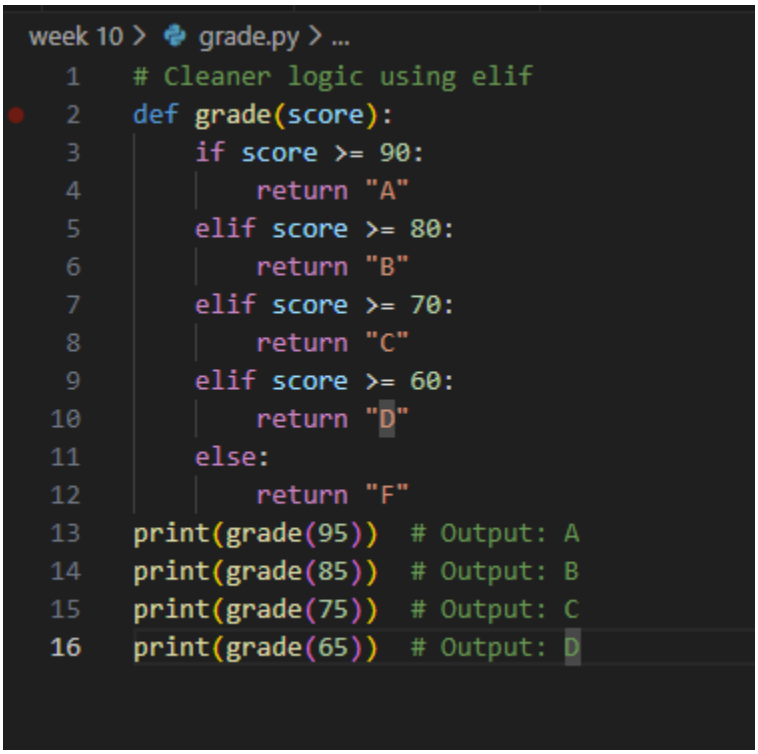
Sample Input Code:

```
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.

>>> Fixed code :



```python
#using dictionary mapping
def grade(score):
    grade_mapping = {
        range(90, 101): "A",
        range(80, 90): "B",
        range(70, 80): "C",
        range(60, 70): "D",
        range(0, 60): "F"
    }
    for key in grade_mapping:
        if score in key:
            return grade_mapping[key]
    return "Invalid Score"
print(grade(95))  # Output: A
print(grade(85))  # Output: B
print(grade(75))  # Output: C
print(grade(65))  # Output: D
print(grade(55))  # Output: F
```

>>> Output :

```
PROBLEMS 1    OUTPUT    DEBUG CONSOLE    T
PS C:\Users\User\OneDrive\Desktop\AIAC>
IAC/week 10/grade.py"
A
B
C
D
A
B
C
D
F
PS C:\Users\User\OneDrive\Desktop\AIAC>
```