AI ASSISTED CODING

ASSIGNMENT-10.1

NAME: K. RUCHITHA HT NO:2303A52069 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.

1.py

Assignment 10.1 > 1.py > ...

```python
# Fix The Below Code With Proper Docstring and Syntax Errors and Proper Function Name
# FIXED: Added docstring and corrected function implementation
def calc_average(marks):
    """Calculate the average score of a student."""
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return average  # FIXED: Corrected typo from "avrage" to "average"

marks = [85, 90, 78, 92]
print("Average Score is ", calc_average(marks))  # FIXED: Added missing closing parenthesis

```

```
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.

2.py

Assignment 10.1 > 2.py > calculate_rectangle_area

```python
# Fix The Below Code With Proper Docstring and Syntax Errors and Proper Understandable Function Name
# Follow PEP 8 Style Guide
# FIXED: Added docstring and corrected function implementation
def calculate_rectangle_area(length, breadth):
    """
    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.
    """
    return length * breadth
print(calculate_rectangle_area(10, 20))
```

```
200
```

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.

3.py

Assignment 10.1 > 3.py > ...

```python
# Refactor the code below make code more readable without changing its logic.

def calculate_percentage(part, whole):
    """Calculate the percentage of a part relative to a whole.
    Args:
        part (float): The part value.
        whole (float): The whole value."""
    return (part * whole) / 100 # Calculates the percentage by multiplying the part with the whole and dividing by 100.
a=200
b=15
print(calculate_percentage(a,b))
```

```
30.0
```

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.

4.py

Assignment 10.1 > 4.py > ...

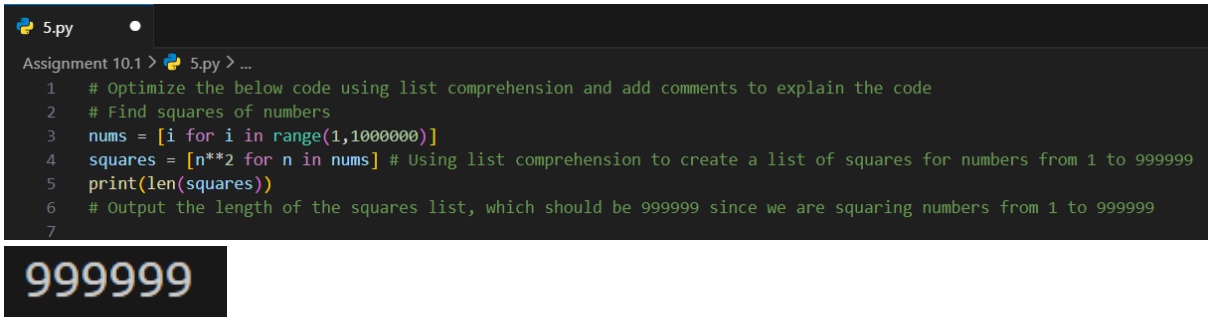
```python
# Refactor the to code break repetitive or long code into reusable functions
students = ["Alice", "Bob", "Charlie"]

def welcome_student(student_name):
    """Welcome a student by name."""
    print("Welcome", student_name)

for student in students:
    welcome_student(student)
```

```
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.



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.

6.py

Assignment 10.1 > 6.py > ...

```python
# Simplify the below code using if-elif-else statements
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"
print(grade(95))  # Output: A
```

```
A
```