

NAME:SYEDA HAFSA FATHIMA

HTNO:2303A52381

BATCH:39

ASSIGNMENT-10.1

TASK 1:

```
# 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.
"""This Python file contains functions for performing basic operations on lists of numbers,
including calculating the average, finding the maximum value, filtering even numbers, and categorizing numbers into positive,
negative, and zero. Each function is designed to handle a list of numerical inputs and return the appropriate results based on the
provided data."""
#write correct code fr above and add comments explaining the fixes made to correct syntax and logic errors.
# Calculate average score of a student
def calc_average(marks):
    """Calculate and return the average score from a list of marks."""
    total = 0
    for m in marks:
        total += m # Indent the line to be inside the loop
    average = total / len(marks) # calculate average after the loop
    return average # Fix typo from 'avrage' to 'average'
marks = [85, 90, 78, 92]
print("Average Score is ", calc_average(marks)) # Add closing parenthesis to complete the print statement
```

TASK 2:

```
# 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.
def area_of_rect(length, breadth):
    """Calculate and return the area of a rectangle given its length and breadth."""
    return length * breadth # Use descriptive variable names and proper spacing
print(area_of_rect(10, 20)) # Add space after comma for better readability
# Task Description #3 - Readability Enhancement
```

TASK 3:

```

# 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.
def calculate_percentage(value, percentage):
    """Calculate and return the percentage of a given value."""
    return value * percentage / 100 # Use descriptive function name and variable names
amount = 200 # Use descriptive variable name for the amount
percent = 15 # Use descriptive variable name for the percentage
print(calculate_percentage(amount, percent)) # Use descriptive function name in the print statement
# Task Description #3 - Readability Enhancement

```

TASK 4:

```

# 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.
def welcome_students(students):
    """Print a welcome message for each student in the list."""
    for student in students: # Loop through each student in the list
        print("Welcome", student) # Print a welcome message for the current student
students = ["Alice", "Bob", "Charlie"]
welcome_students(students) # Call the function to welcome all students

```

TASK 5:

```

# 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.
# Find squares of numbers
nums = [i for i in range(1, 1000000)]
# Use list comprehension to calculate squares in a more efficient way
squares = [n**2 for n in nums] # This is more efficient than appending in a loop
print(len(squares)) # Print the length of the squares list to confirm it contains the correct number of elements

```

TASK 6:

```
# 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.
def grade(score):
    """Return the letter grade corresponding to the given score."""
    if score >= 90:
        return "A"
    elif score >= 80: # Use elif to simplify the logic
        return "B"
    elif score >= 70: # Use elif to simplify the logic
        return "C"
    elif score >= 60: # Use elif to simplify the logic
        return "D"
    else:
        return "F" # Return "F" for scores below 60
```