ASSIGNMENT-10.5

2303A510G2

UDAY CH

Batch-11

Lab Objectives

- Use AI for automated code review and quality enhancement. •
  Identify and fix syntax, logical, performance, and security issues in
  Python code.
- Improve readability and maintainability through structured
  refactoring and comments.
- Apply prompt engineering for targeted improvements.
- Evaluate AI-generated suggestions against PEP 8 standards and
  software engineering best practices

Lab Outcomes

1. Students will be able to use AI tools to review code.
2. Students will be able to improve code quality and readability. 3. Students will be
   able to identify and fix common coding issues.

ask Description #1 – Variable Naming Issues Task:

Use AI to improve unclear variable names.

Sample Input Code:

```
def f(a, b):
return a + b
print(f(10, 20))
```

Problem with the above code is:

Task Description #1 – Variable Naming Issues

The error occurs because the function f is defined to take two parameters (a and b), but only one argument (2.3) is provided when calling the function.

Corrected Code:

```
#  give the correct code and explain it in line comments
def f(a, b):
    return a + b  # This function takes two parameters, a and b, and returns their sum.
print(f(2.3, 3.7))  # This calls the function f with two arguments, 2.3 and 3.7, and prints the result, which is 6.0.
```

Task Description #2 – Missing Error Handling

Task: Use AI to add proper error handling.

Sample Input Code:

def divide(a, b): return

a / b print(divide(10,

0))

Problem with the above code :

The error in the code above is a ZeroDivisionError. This occurs because the function is trying to divide the number 10 by 0, which is mathematically undefined. In Python, attempting to divide by zero will raise this error, indicating that the operation cannot be performed. To fix this error, you should add a check to ensure that the denominator (b) is not zero before performing the division.

Corrected code with the well **documented** and well **drafted** and **optimized** Code:

```
# give the corrected code with well documented and well drafted and optimized Code:
def divide(a, b):
    # Check if the denominator is zero to avoid division by zero error
    if b == 0:
        return "Error: Division by zero is not allowed."
    else:
        return a / b  # Perform the division and return the result
print(divide(10, 0))  # This will now return an error message instead of crashing the program
print(divide(10, 2))  # This will return 5.0,
```

Task Description #3: Student Marks Processing System The

following program calculates total, average, and grade of a

student, but it has poor readability, style issues, and no error

handling.

marks=[78,85,90,66,88]

t=0 for i in marks: t=t+i

a=t/len(marks)

if a>=90:

print("A") elif

a>=75:

print("B") elif

a>=60:

print("C") else:

print("F")

Explanation for the code (error):

The error in the above code is due to incorrect indentation. In Python, indentation is crucial for defining the scope of loops and conditional statements. The lines inside the for loop and the if-elifelse statements should be indented properly to indicate that they belong to those blocks.

Corrected code with the well **documented** and well **drafted** and **optimized** Code:

```
# Corrected code with the well documented and well drafted and optimized Code: give
# Create a list of marks obtained by students
marks = [78, 85, 90, 66, 88]
# Initialize a variable to store the total marks
total_marks = 0
# Iterate through each mark in the list and add it to the total
for mark in marks:
    total_marks += mark  # Add the current mark to the total
# Calculate the average marks by dividing the total marks by the number of students
average_marks = total_marks / len(marks)  # Calculate the average
# Determine the grade based on the average marks
if average_marks >= 90:
    print("Grade: A")  # Print Grade A for average marks 90 or above
elif average_marks >= 75:
    print("Grade: B")  # Print Grade B for average marks between 75 and 89
elif average_marks >= 60:
    print("Grade: C")  # Print Grade C for average marks between 60 and 74
else:
    print("Grade: F")  # Print Grade F for average marks below 60
```

Task Description #4: Use AI to add docstrings and inline comments to

the following function.

def factorial(n):

result = 1 for i in

range(1,n+1):

result *= i return

result

Explanation for the above code :

The error in the above code is that the lines inside the function are not indented properly. In Python, indentation is crucial as it defines the scope of loops, functions, and other code blocks. The lines that calculate the factorial should be indented to indicate that they are part of the function. Here is the corrected code:

Corrected code with the well **documented** and well **drafted** and **optimized** Code:

```python
# Corrected code with the well documented and well drafted and optimized Code
def factorial(n):
    # Initialize result to 1, as the factorial of 0 is 1
    result = 1

    # Loop from 1 to n and multiply the result by each number
    for i in range(1, n + 1):
        result *= i

    return result
# Test the function with some numbers
print(factorial(5))  # Output: 120, because 5! = 5 * 4 * 3 * 2 * 1 = 120
print(factorial(0))  # Output: 1, because 0! = 1 by definition
print(factorial(1))  # Output: 1, because 1! = 1 by definition
```

Output:

```
120
1
1
```

Task Description #5: Password Validation System (Enhanced) The

following Python program validates a password using only a

minimum length check, which is insufficient for real-world

security requirements.

pwd = input("Enter password: ")

if len(pwd) >= 8: print("Strong")

else: print("Weak")

Task:

1. Enhance password validation using AI assistance to include

multiple security rules such as: o Minimum length

requirement

o Presence of at least one uppercase letter

o Presence of at least one lowercase letter o Presence of at least one

digit o Presence of at least one special character

2. Refactor the program to: o Use meaningful variable

and function names o Follow PEP 8 coding standards o

Include inline comments and a docstring

3. Analyze the improvements by comparing the

original and AI- enhanced versions in terms of:

Explanation for the above code (Error):

The error in the code is due to incorrect indentation. In Python, indentation is crucial for defining the scope of loops, functions, and conditional statements. The lines following the 'if' and 'else' statements should be indented to indicate that they belong to those blocks. The corrected code should look like this:

Corrected code with the well **documented** and well **drafted** and **optimized** Code:

```
# Corrected code with the well documented and well drafted and optimized Code:
# Get the password input from the user
pwd = input("Enter password: ")
# Check the length of the password
if len(pwd) >= 8:
    print("Strong")  # If the password is 8 characters or more, it's strong
else:
    print("Weak")  # If the password is less than 8 characters, it's weak
```

Output:

```
Enter password: kumar
Weak
```