

Lab Assignment-10.5

Name: Goutham Reddy

Hall Ticket no: 2303A51345

Batch No: 20

Task -1:

Prompt:

Task: Use AI to improve unclear variable names.

Sample Input Code:

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

Expected Output:

- Code rewritten with meaningful function and variable names.

```
#Task-1  
def add_numbers(first_number, second_number):  
    return first_number + second_number  
print(add_numbers(10, 20))
```

OUTPUT:

```
PS C:\Users\temba\OneDrive\Desktop\Ai_assig_coding> & C:\Users\temba\AppData\Local\Programs\Python\Python38-64\python.exe C:\Users\temba\OneDrive\Desktop\Ai_assig_coding\10.5-Assignment.py  
c:\Users\temba\OneDrive\Desktop\Ai_assig_coding\10.5-Assignment.py:122: SyntaxWarning: "in the future. Did you mean "\d"? A raw string is also an option.  
    if not re.search(r'\d', password):  
30
```

Justification:

The refactored code improves readability by using descriptive names. The function name clearly indicates its purpose. Adding a docstring improves documentation. The updated version follows PEP 8 naming conventions and enhances maintainability.

Task 2:

Prompt:

ask 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)) Expected Output:
```

- Code with exception handling and clear error messages

```
#Task-2

def divide(a, b):
    try:
        if b == 0:
            raise ValueError("Error: Division by zero is not allowed. Please provide a non-zero divisor.")
        result = a / b
        return result
    except ValueError as ve:
        print(f"ValueError: {ve}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

print(divide(10, 0))
print(divide(10, 2))
```

Output:

```
PS C:\Users\temba\OneDrive\Desktop\Ai_assig_coding> & C:\Users\temba\AppData\Local\Programs\Python\Python39\python.exe C:\Users\temba\OneDrive\Desktop\Ai_assig_coding\10.5-Assignment.py
c:\Users\temba\OneDrive\Desktop\Ai_assig_coding\10.5-Assignment.py:121: SyntaxWarning: "\d" is an invalid escape sequence
  if not re.search(r'\d', password):
ValueError: Error: Division by zero is not allowed. Please provide a non-zero divisor.
None
5.0
PS C:\Users\temba\OneDrive\Desktop\Ai_assig_coding>
```

Justification: The AI-enhanced version prevents program crashes by adding exception handling. It provides meaningful error messages instead of terminating execution. This improves robustness and user experience.

Task-3:

Prompt:

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") Task:

- Use AI to refactor the code to follow PEP 8 standards.
- Add meaningful variable names, functions, and comments.
- Add basic input validation and documentation.

```
28 #Task-3
29
30 def calculate_student_grade(marks):
31     if not isinstance(marks, list):
32         raise TypeError("Marks must be a list")
33
34     if len(marks) == 0:
35         raise ValueError("Marks list cannot be empty")
36
37     try:
38         # Validate that all marks are numeric
39         for mark in marks:
40             if not isinstance(mark, (int, float)):
41                 raise TypeError(f"Mark must be numeric, got {type(mark).__name__}")
42
43         total = sum(marks)
44         average = total / len(marks)
45
46         # Determine grade based on average
47         if average >= 90:
48             grade = "A"
49         elif average >= 75:
50             grade = "B"
51         elif average >= 60:
52             grade = "C"
53         else:
54             grade = "F"
55
56         return total, average, grade
57
58     except Exception as e:
59         print(f"Error processing marks: {e}")
60         return None
61
62 # Test the function
63 marks = [78, 85, 90, 66, 88]
64 result = calculate_student_grade(marks)
65 if result:
66     total, average, grade = result
67     print(f"Total: {total}, Average: {average:.2f}, Grade: {grade}")
68 ...
69
```

Output:

```
PS C:\Users\temba\OneDrive\Desktop\Ai_assig_coding> & C:\Users\temba\AppData\Local\Programs\Python\Python38\python.exe Desktop\Ai_assig_coding\10.5-Assignment.py
c:\Users\temba\OneDrive\Desktop\Ai_assig_coding\10.5-Assignment.py:108: SyntaxWarning: "\d" is an invalid escape sequence
  if not re.search(r'\d', password):
not work in the future. Did you mean "\\d"? A raw string is also an option.
Total: 407, Average: 81.40, Grade: B
PS C:\Users\temba\OneDrive\Desktop\Ai_assig_coding>
```

Justification:

The AI refactored the code into a structured function, improving modularity and reusability. Meaningful variable names increase clarity. Input validation ensures safety. The code now follows PEP 8 guidelines and improves maintainability.

Task 4:

Prompt:

ask 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

```
#Task-4

def factorial(n):
    """
    Calculate the factorial of a given non-negative integer.
    """
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

# Calling the function
num = 5
print("Factorial of", num, "is", factorial(num))
```

Output:

```
PS C:\Users\temba\OneDrive\Desktop\Ai_assig_coding> & C:\Users\temba\AppData\Local\Microsoft\Windows\Shell\10.5-Assignment.py
c:\Users\temba\OneDrive\Desktop\Ai_assig_coding\10.5-Assignment.py:103: SyntaxWarning: 'd' character is not valid in a string literal
  if not re.search(r'\d', password):
  ~~~~~
Factorial of 5 is 120
PS C:\Users\temba\OneDrive\Desktop\Ai_assig_coding>
```

Justification:

The addition of docstrings and inline comments improves readability and documentation quality. The function now explains its purpose, parameters, and return values clearly.

Task 5:

Prompt:

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:
 - o Code readability and structure
 - o Maintainability and reusability
 - o Security strength and robustness
4. Justify the AI-generated changes, explaining why each added rule and refactoring decision improves the overall quality of the program.

```
84 #Task-5
85 import re
86 def validate_password(password):
87     feedback = []
88     # Check minimum length
89     if len(password) < 8:
90         feedback.append("Password must be at least 8 characters long")
91     # Check for uppercase letter
92     if not re.search(r'[A-Z]', password):
93         feedback.append("Password must contain at least one uppercase letter")
94     # Check for lowercase letter
95     if not re.search(r'[a-z]', password):
96         feedback.append("Password must contain at least one lowercase letter")
97     # Check for digit
98     if not re.search(r'\d', password):
99         feedback.append("Password must contain at least one digit")
100     # Check for special character
101     if not re.search(r'!@#%&*()_+~\[\]\{\};\':",.<>?/\|\'~]', password):
102         feedback.append("Password must contain at least one special character")
103     is_valid = len(feedback) == 0
104     return is_valid, feedback
```

```
104     return is_valid, feedback
105 # Main program
106 password = input("Enter password: ")
107 is_strong, issues = validate_password(password)
108 if is_strong:
109     print("Strong")
110 else:
111     print("Weak")
112     for issue in issues:
113         print(f" - {issue}")
```

Output:

```
PS C:\Users\temba\OneDrive\Desktop\Ai_assig_coding> & C:\Users\temba\AppData\Local\Microsoft\Windows\Terminal\10.5-Assignment.py
Enter password: Sru#1234554321
Strong
```

Justification:

The enhanced version significantly improves security by enforcing multiple validation rules. The use of regular expressions increases flexibility and maintainability. The program is modular, readable, and follows PEP 8 standards. Compared to the original version, the AI-enhanced version provides stronger validation, improved structure, and better real-world applicability.

