Lab Assignment-10.5

Name: A.Shiva Spandana

Hall Ticket no: 2303A51320

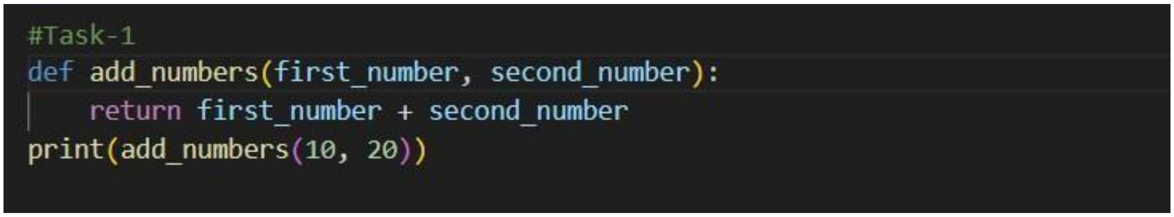
Batch No: 19

Task -1:

Prompt:

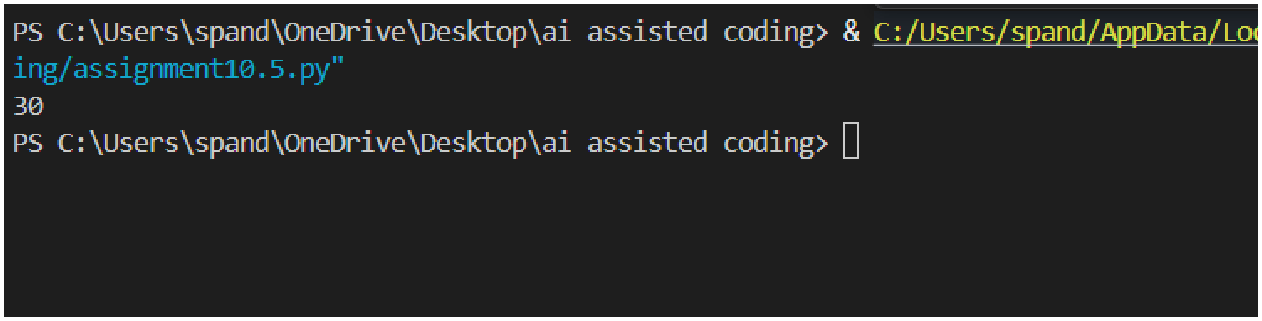
Task: Use AI to improve unclear variable names.
Sample Input Code: def f(a, b): return a
+ b print(f(10, 20)) Expected
Output:

• Code rewritten with meaningful function and variable names.



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

OUTPUT:



```
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding> & C:/Users/spand/AppData/Lo
ing/assignment10.5.py"
30
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding> 
```

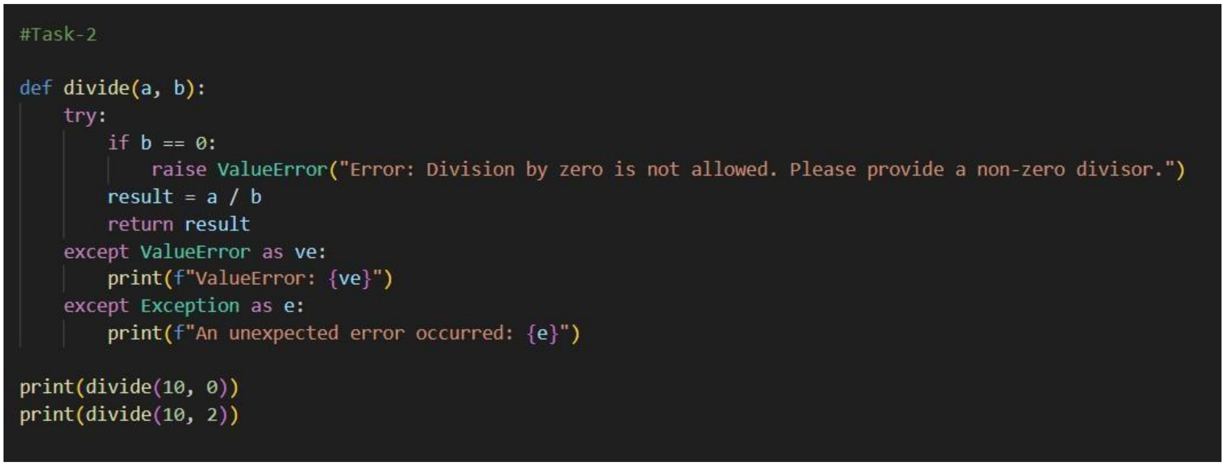
Justification:

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:

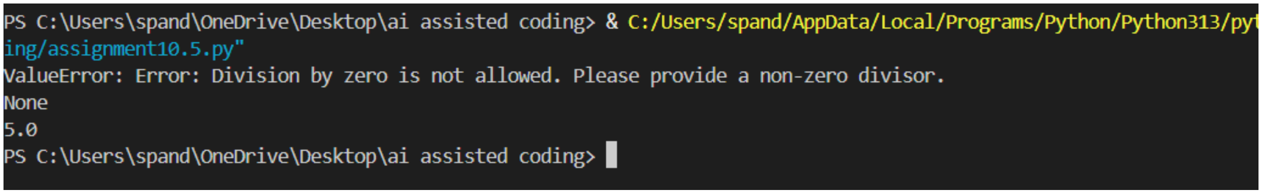
Prompt:

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\spand\OneDrive\Desktop\ai assisted coding> & C:/Users/spand/AppData/Local/Programs/Python/Python313/pyt
ing/assignment10.5.py"
ValueError: Error: Division by zero is not allowed. Please provide a non-zero divisor.
None
5.0
PS C:\Users\spand\OneDrive\Desktop\ai assisted 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:

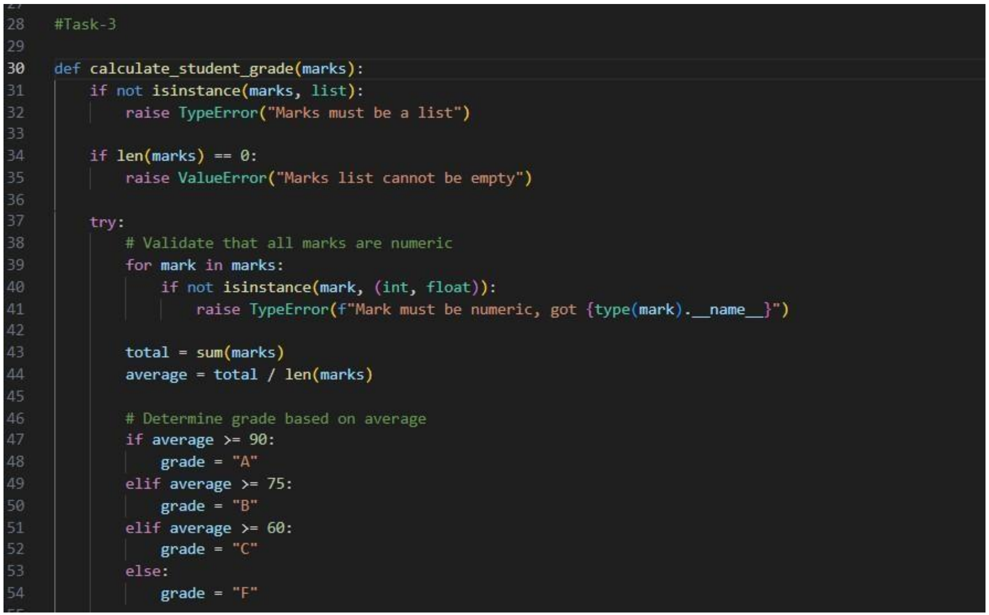
print("A") elif a>=75:

print("B") elif a>=60:

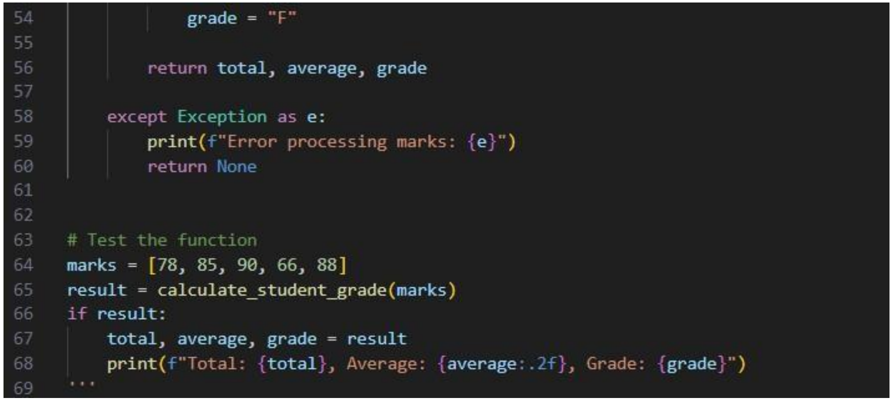
print("C") else:

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.

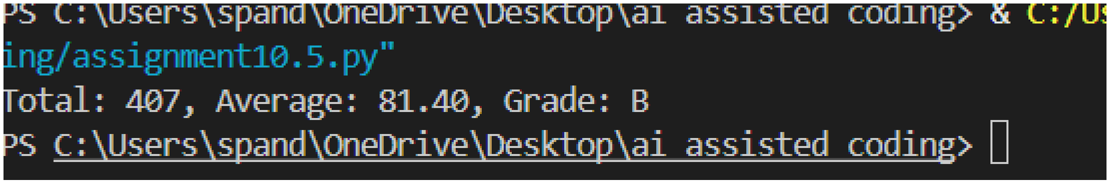


```python
#Task-3

def calculate_student_grade(marks):
    if not isinstance(marks, list):
        raise TypeError("Marks must be a list")

    if len(marks) == 0:
        raise ValueError("Marks list cannot be empty")

    try:
        # Validate that all marks are numeric
        for mark in marks:
            if not isinstance(mark, (int, float)):
                raise TypeError(f"Mark must be numeric, got {type(mark).__name__}")

        total = sum(marks)
        average = total / len(marks)

        # Determine grade based on average
        if average >= 90:
            grade = "A"
        elif average >= 75:
            grade = "B"
        elif average >= 60:
            grade = "C"
        else:
            grade = "F"
```



```python
            grade = "F"

        return total, average, grade

    except Exception as e:
        print(f"Error processing marks: {e}")
        return None


# Test the function
marks = [78, 85, 90, 66, 88]
result = calculate_student_grade(marks)
if result:
    total, average, grade = result
    print(f"Total: {total}, Average: {average:.2f}, Grade: {grade}")
'''
```

Output:



```
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding> & C:/Us
ing/assignment10.5.py"
Total: 407, Average: 81.40, Grade: B
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding>
```

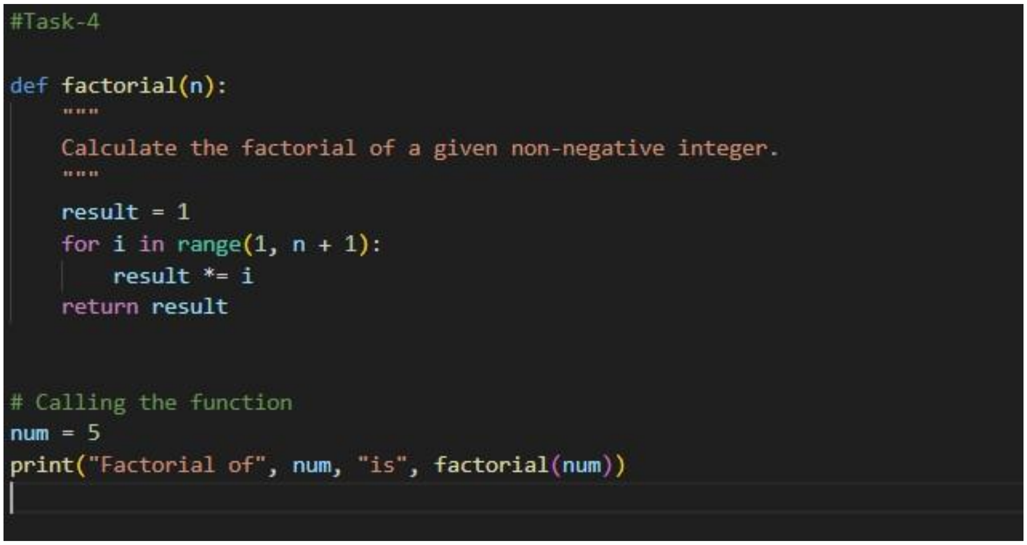
Justification:

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:

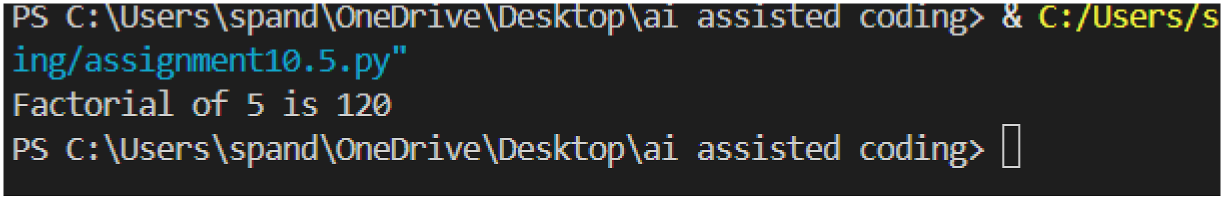
Prompt:

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\spand\OneDrive\Desktop\ai assisted coding> & C:/Users/s
ing/assignment10.5.py"
Factorial of 5 is 120
PS C:\Users\spand\OneDrive\Desktop\ai assisted 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.

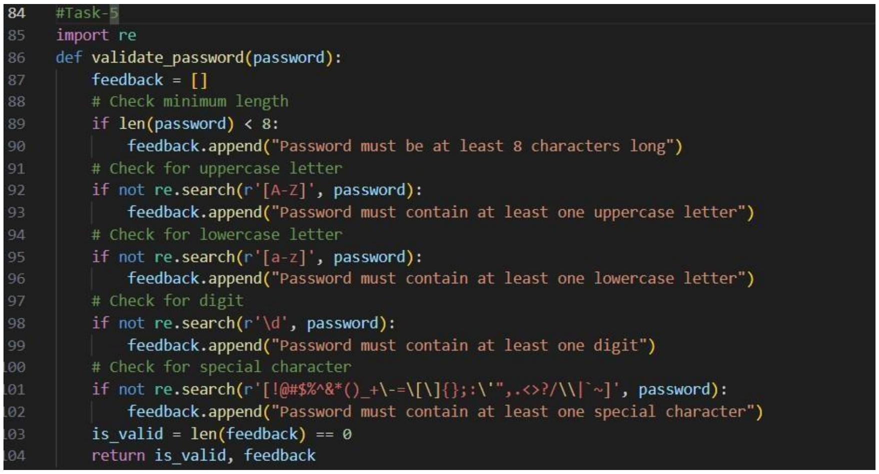
pwd = input("Enter password: ") if
len(pwd) >= 8: print("Strong")
else: print("Weak") Task:

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:

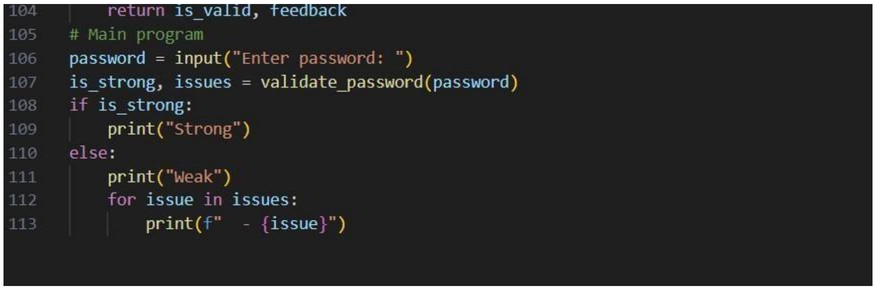
o Use meaningful variable and function names

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.

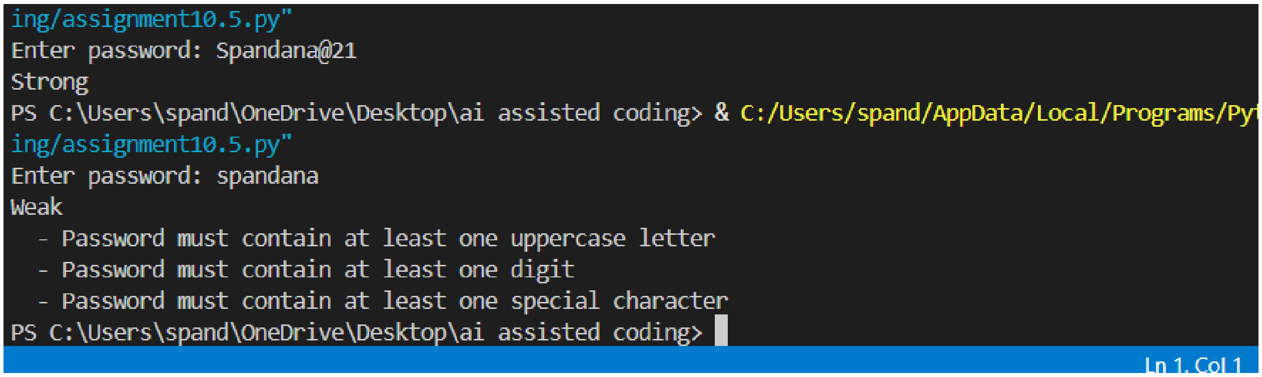


```
#Task-5
import re
def validate_password(password):
    feedback = []
    # Check minimum length
    if len(password) < 8:
        feedback.append("Password must be at least 8 characters long")
    # Check for uppercase letter
    if not re.search(r'[A-Z]', password):
        feedback.append("Password must contain at least one uppercase letter")
    # Check for lowercase letter
    if not re.search(r'[a-z]', password):
        feedback.append("Password must contain at least one lowercase letter")
    # Check for digit
    if not re.search(r'\d', password):
        feedback.append("Password must contain at least one digit")
    # Check for special character
    if not re.search(r'[!@#$%^&*()_+\-=\[\]{};:\'",.<>?/\\|`~]', password):
        feedback.append("Password must contain at least one special character")
    is_valid = len(feedback) == 0
    return is_valid, feedback
```



```
    return is_valid, feedback
# Main program
password = input("Enter password: ")
is_strong, issues = validate_password(password)
if is_strong:
    print("Strong")
else:
    print("Weak")
    for issue in issues:
        print(f"  - {issue}")
```

Output:



```
ing/assignment10.5.py"
Enter password: Spandana@21
Strong
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding> & C:/Users/spand/AppData/Local/Programs/Py
ing/assignment10.5.py"
Enter password: spandana
Weak
  - Password must contain at least one uppercase letter
  - Password must contain at least one digit
  - Password must contain at least one special character
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding>
```

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