SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name	Dr. Rishabh Mittal		
Instructor(s) Name	Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju		
CourseCode	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week5 – Friday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches
Assignment Number: 10.5 (Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	**Lab 10 – Code Review and Quality: Using AI to Improve Code Quality and Readability**	Week5 - Friday

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.

Task 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))
```

Expected Output:

• Code rewritten with meaningful function and variable names.

Prompt

Refactor the given function by replacing unclear names with meaningful names and follow PEP 8 standards.

Code

```python
def add_numbers(first_number, second_number):
    return first_number + second_number
print(add_numbers(10, 20))

def subtract_numbers(first_number, second_number):
    return first_number - second_number
print(subtract_numbers(10, 20))
def multiply_numbers(first_number, second_number):
    return first_number * second_number
print(multiply_numbers(10, 20))
```

Output

```
PS C:\Users\konda\Downloads\AI Assistant> & \\?\C:\Users\konda\Ap
"
30
-10
200
PS C:\Users\konda\Downloads\AI Assistant>
```

Explanation

The function name f and variables a, b were unclear.

AI replaced them with meaningful names, improving readability and maintainability.

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))
```

Expected Output:

• Code with exception handling and clear error messages

Prompt

Improve the divide function by adding exception handling to prevent runtime errors.

Code

```python
def divide_numbers(num1, num2):
    try:
        return num1 / num2
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed"


print(divide_numbers(10, 0))
print(divide_numbers(10, 2))

```

Output

```
PS C:\Users\konda\Downloads\AI Assistant> & \\?\C:\Users\konda\AppData\Local\Mi
"
Error: Division by zero is not allowed
5.0
PS C:\Users\konda\Downloads\AI Assistant>
```

Explanation

AI added a try-except block to handle division by zero, preventing program crashes and improving reliability.

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.

Prompt

Refactor the student marks program to follow PEP 8, add functions, meaningful variables, and validation.

Code

day3.py > ...

```python
def calculate_student_grade(marks):
    if not marks:
        return "Invalid marks list"

    total_marks = sum(marks)
    average_marks = total_marks / len(marks)

    if average_marks >= 90:
        grade = "A"
    elif average_marks >= 75:
        grade = "B"
    elif average_marks >= 60:
        grade = "C"
    else:
        grade = "F"

    return total_marks, average_marks, grade


marks_list = [78, 85, 90, 66, 88]
result = calculate_student_grade(marks_list)

print("Total:", result[0])
print("Average:", result[1])
print("Grade:", result[2])
```

Output

```
PS C:\Users\konda\Downloads\AI Assistant> & \\?\C:\Users\konda\Appl
"
Total: 407
Average: 81.4
Grade: B
PS C:\Users\konda\Downloads\AI Assistant>
```

Explanation

AI improvements:

- Meaningful names
- Function-based modular design
- Used built-in sum() for efficiency
- Added validation for empty list

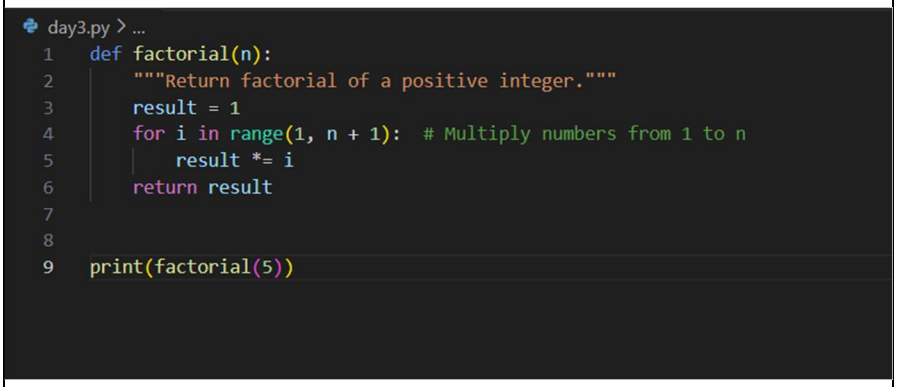
Task Description #4: Use AI to add docstrings and inline comments to the following function.

```
def factorial(n):
    result = 1
    for i in range(1,n+1):
        result *= i
    return result
```

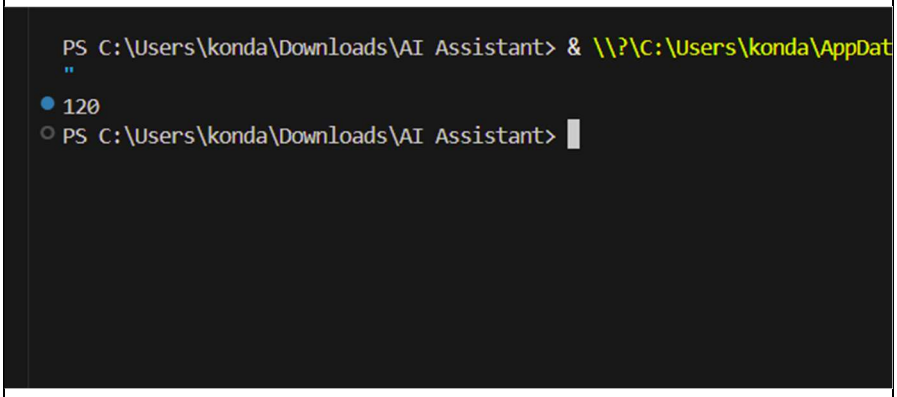
Prompt

Add docstring and inline comments to factorial function.

Code



Output



Explanation

AI added documentation and comments to explain functionality, improving clarity and maintainability.

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:
    - Minimum length requirement
    - Presence of at least one uppercase letter
    - Presence of at least one lowercase letter
    - Presence of at least one digit
    - Presence of at least one special character
2. Refactor the program to:
    - Use meaningful variable and function names
    - Follow PEP 8 coding standards
    - Include inline comments and a docstring
3. Analyze the improvements by comparing the original and AI-enhanced versions in terms of:
    - Code readability and structure
    - Maintainability and reusability
    - 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.

Prompt

Enhance password validation using multiple security rules and refactor code with functions and comments.

Code

```python
import re
def check_password_strength(password):
    if len(password) < 8:
        return "Weak Password"

    if not re.search(r"[A-Z]", password):
        return "Weak: Add uppercase letter"

    if not re.search(r"[a-z]", password):
        return "Weak: Add lowercase letter"

    if not re.search(r"[0-9]", password):
        return "Weak: Add digit"

    if not re.search(r"[!@#$%^&*]", password):
        return "Weak: Add special character"

    return "Strong Password"


user_password = input("Enter password: ")
print(check_password_strength(user_password))
```

Output

```
PS C:\Users\konda\Downloads\AI Assistant> & \\?\C:\Users\konda\AppData\Local\
"
Enter password: Shiva@143
Strong Password
PS C:\Users\konda\Downloads\AI Assistant>
```

Explanation

AI added multiple security checks (uppercase, lowercase, digit, special character).

This improves password strength, security, and real-world usability.