

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		<div> <div>Mr. S Naresh Kumar</div> <div>Ms. B. Swathi</div> <div>Dr. Sasanko Shekhar Gantayat</div> <div>Mr. Md Sallauddin</div> <div>Dr. Mathivanan</div> <div>Mr. Y Srikanth</div> <div>Ms. N Shilpa</div> <div>Dr. Rishabh Mittal (Coordinator)</div> <div>Dr. R. Prashant Kumar</div> <div>Mr. Ankushavali MD</div> <div>Mr. B Viswanath</div> <div>Ms. Sujitha Reddy</div> <div>Ms. A. Anitha</div> <div>Ms. M.Madhuri</div> <div>Ms. Katherashala Swetha</div> <div>Ms. Velpula sumalatha</div> <div>Mr. Bingi Raju</div> </div>	
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	

	<p>Lab Objectives</p> <ul style="list-style-type: none">• 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 <p>Lab Outcomes</p> <ol style="list-style-type: none">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.	
	<p>Task Description #1 – Variable Naming Issues Task:</p> <p>Use AI to improve unclear variable names.</p> <p>Sample Input Code:</p> <pre>def f(a, b): return a + b print(f(10, 20))</pre> <p>Expected Output:</p> <ul style="list-style-type: none">• Code rewritten with meaningful function and variable names. <p>Prompt</p> <p>Refactor the given function by replacing unclear names with meaningful names and follow PEP 8 standards.</p>	

Code

```
day3.py > ...
1 def add_numbers(first_number, second_number):
2     return first_number + second_number
3 print(add_numbers(10, 20))
4
5 def subtract_numbers(first_number, second_number):
6     return first_number - second_number
7 print(subtract_numbers(10, 20))
8 def multiply_numbers(first_number, second_number):
9     return first_number * second_number
10 print(multiply_numbers(10, 20))
11
12
13
14
```

Output

```
0.5
PS C:\Users\konda\Downloads\AI Assistant> & \\?C:\Users\konda\AppData\Local\Microsoft\Windows\AI Assistant\AI Assistant.exe
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

```
day3.py > ...
1  def divide_numbers(num1, num2):
2      try:
3          return num1 / num2
4      except ZeroDivisionError:
5          return "Error: Division by zero is not allowed"
6
7
8  print(divide_numbers(10, 0))
9  print(divide_numbers(10, 2))
10
11
```

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
```

	<pre>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")</pre> <p>Task:</p> <ul style="list-style-type: none">• Use AI to refactor the code to follow PEP 8 standards.• Add meaningful variable names, functions, and comments.• Add basic input validation and documentation. <p>Prompt</p> <p>Refactor the student marks program to follow PEP 8, add functions, meaningful variables, and validation.</p> <p>Code</p>	
--	--	--

```

1 def calculate_student_grade(marks):
2     if not marks:
3         return "Invalid marks list"
4
5     total_marks = sum(marks)
6     average_marks = total_marks / len(marks)
7
8     if average_marks >= 90:
9         grade = "A"
10    elif average_marks >= 75:
11        grade = "B"
12    elif average_marks >= 60:
13        grade = "C"
14    else:
15        grade = "F"
16
17    return total_marks, average_marks, grade
18
19
20 marks_list = [78, 85, 90, 66, 88]
21 result = calculate_student_grade(marks_list)
22
23 print("Total:", result[0])
24 print("Average:", result[1])
25 print("Grade:", result[2])

```

Output

```

PS C:\Users\konda\Downloads\AI Assistant> & \\?\C:\Users\konda\AppData\Local\Microsoft\Windows\Apps\AI Assistant.exe
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

```
day3.py > ...  
1  def factorial(n):  
2      """Return factorial of a positive integer."""  
3      result = 1  
4      for i in range(1, n + 1): # Multiply numbers from 1 to n  
5          result *= i  
6      return result  
7  
8  
9  print(factorial(5))
```

Output

```
PS C:\Users\konda\Downloads\AI Assistant> & \\?\C:\Users\konda\AppData  
..  
120  
PS C:\Users\konda\Downloads\AI Assistant> |
```

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

	<p>else: minimum length check, which is insufficient for real-world password requirements.</p> <p>Task: password = input("Enter password: ")</p> <ol style="list-style-type: none"> if len(password) >= 8: <p>print("Strong")</p> <p>print("Weak")</p> <p>Enhance password validation using AI assistance to include multiple security rules such as:</p> <ul style="list-style-type: none"> 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 <ol style="list-style-type: none"> Refactor the program to: <ul style="list-style-type: none"> Use meaningful variable and function names Follow PEP 8 coding standards Include inline comments and a docstring Analyze the improvements by comparing the original and AI-enhanced versions in terms of: <ul style="list-style-type: none"> Code readability and structure Maintainability and reusability Security strength and robustness 	
--	---	--

	<p>4. Justify the AI-generated changes, explaining why each added rule and refactoring decision improves the overall quality of the program. ompt</p>	
--	--	--

Prance password validation using multiple security rules and .ctor code with functions and comments.

Code

```
day3.py
day3.py 2 ...
1 import re
2 def check_password_strength(password):
3     if len(password) < 8:
4         return "Weak Password"
5
6     if not re.search(r"[A-Z]", password):
7         return "Weak: Add uppercase letter"
8
9     if not re.search(r"[a-z]", password):
10        return "Weak: Add lowercase letter"
11
12    if not re.search(r"[0-9]", password):
13        return "Weak: Add digit"
14
15    if not re.search(r"[!@#$%^&*]", password):
16        return "Weak: Add special character"
17
18    return "Strong Password"
19
20
21 user_password = input("Enter password: ")
22 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.