NAME:SIRIVELLA SANJANA ID:2403A510D4 BATCH:05

SCHOOLOFCOMPUTERSCIENCEANDARTIFICIAL INTELLIGENCE		**DEPARTMENTOFCOMPUTERSCIENCE ENGINEERING**	
ProgramName:B.Tech		**AssignmentType:Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName STUDENT DETAILS:	VenkataramanaVeeramsetty. Sirivella sanjana Batch-05 2403a510d4		
Instructor(s)Name	Dr.V.Venkataramana(Co-Ordinator) Dr.T.SampathKumar Dr.PramodaPatro Dr.BrijKishor Tiwari Dr.J.Ravichander Dr.MohammandAliShaik Dr.AnirodhKumar Mr.S.Naresh Kumar Dr.RAJESHVELPULA Mr.KundhanKumar Ms.Ch.Rajitha Mr.MPrakash Mr.B.Raju Intern1(Dharmateja) Intern2(Sai Prasad) Intern3(Sowmya) NS_2(Mounika)		
CourseCode	24CS002PC215	**CourseTitle**	AIAssistedCoding
Year/Sem	II/I	**Regulation**	R24
DateandDay of Assignment	Week1- Monday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	24CSBTB01To 24CSBTB39
AssignmentNumber:2.1(Presentassignmentnumber)/**24**(Totalnumberofassignments)			

Q.No.	Question	*Expected Time to complete*
1	Lab 1: Environment Setup – GitHub Copilot and VS Code Integration Lab Objectives: ● To install and configure GitHub Copilot in Visual Studio Code. To explore AI-assisted code generation using GitHub Copilot.	Week1-Monday

● To analyze the accuracy and effectiveness of Copilot's code suggestions.
● To understand prompt-based programming using comments and code context

Lab Outcomes (LOs):

After completing this lab, students will be able to:

● Set up GitHub Copilot in VS Code successfully.
● Use inline comments and context to generate code with Copilot.
● Evaluate AI-generated code for correctness and readability.
● Compare code suggestions based on different prompts and programming styles.

Task 0

● Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

Expected Output

● Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

Task 1: Factorial without Functions

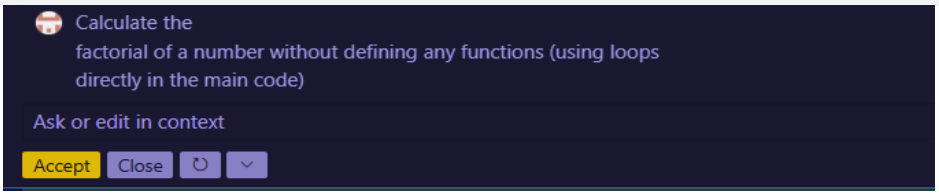
● Description:

Use GitHub Copilot to generate a Python program that calculates the factorial of a number without defining any functions (using loops directly in the main code).

● Expected Output:

○ A working program that correctly calculates the factorial for user-provided input.
○ Screenshots of the code generation process.

Prompt:



Code:

```
# Calculate factorial of a number using a loop (no function

n = int(input("Enter a number: "))
fact = 1

for i in range(1, n + 1):
    fact *= i

print("Factorial of", n, "is", fact)
```

Output:

```
C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab_1>python 1.1.py
Enter a number: 1
Factorial of 1 is 1

C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab_1>python 1.1.py
Enter a number: 9
Factorial of 9 is 362880
```

Task 2: Improving Efficiency

- Description:

Examine the Copilot-generated code from Task 1 and demonstrate how its efficiency can be improved (e.g., removing unnecessary variables, optimizing loops).

- Expected Output:
  - Original and improved versions of the code.
  - Explanation of how the improvements enhance performance

prompt and code:

```python
# Calculate factorial of a number using a loop (no functions
# Task1
n = int(input("Enter a number: "))
fact = 1

for i in range(1, n + 1):
    fact *= i

print("Factorial of", n, "is", fact)

# Calculate factorial of a number efficiently using math.fac
# Improved code(Task2)
import math

n = int(input("Enter a number: "))
fact = math.factorial(n)

print("Factorial of", n, "is", fact)
```

The loop-based factorial calculation is already quite efficient for small numbers. However, you can improve efficiency sligh `math.factorial()` function, which is implemented in C and optimized for performance. If you must use a loop and no fu using local variables and avoiding unnecessary operations.

Task 3: Factorial with Functions

● Description:
Use GitHub Copilot to generate a Python program that calculates the factorial of a number using a user-defined function.
● Expected Output:
○ Correctly working factorial function with sample outputs.
○ Documentation of the steps Copilot followed to generate the function.

Prompt and code:

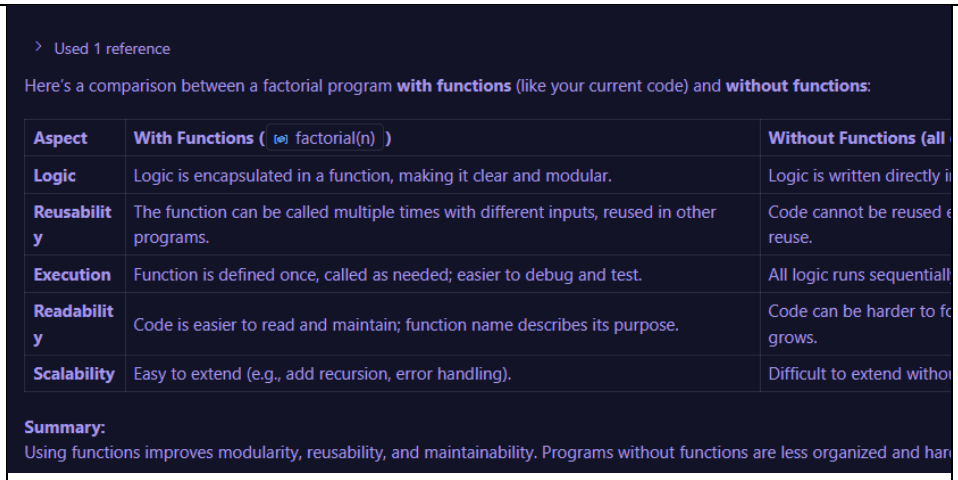
```
# Calculate factorial of a number using a user-defined fun


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


n = int(input("Enter a number: "))
fact = factorial(n)
print("Factorial of", n, "is", fact)
```

```
C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab_1>python 1_1.
Enter a number: 4
Factorial of 4 is 24
```

Task 4: Comparative Analysis – With vs Without Functions

● Description:
Differentiate between the Copilot-generated factorial program with functions and without functions in terms of logic, reusability, and execution.
● Expected Output:
○ A comparison table or short report explaining the differences

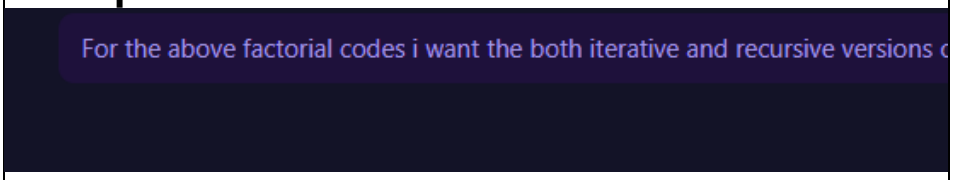
Differentiate between the Copilot-generated factorial program with functions and without functions in terms of logic,
• Expected Output:
o A comparison table or short report explaining the differences



Task 5: Iterative vs Recursive Factorial

● Description:
Prompt GitHub Copilot to generate both iterative and recursive versions of the factorial function.
● Expected Output:
○ Two correct implementations.
○ A documented comparison of logic, performance, and execution flow between iterative and recursive approaches.

Prompt:



Code:

```python
# Calculate factorial of a number using user-defined funct
def factorial_iterative(n):
    fact = 1
    for i in range(1, n + 1):
        fact *= i
    return fact


def factorial_recursive(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial_recursive(n - 1)


n = int(input("Enter a number: "))

print("Iterative: Factorial of", n, "is", factorial_iterat
print("Recursive: Factorial of", n, "is", factorial_recurs
```

Output:

```
C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab_1>python 1_1.
Enter a number: 8
Iterative: Factorial of 8 is 40320
Recursive: Factorial of 8 is 40320

C:\Users\venub\OneDrive\Desktop\AIAC_Lab\Lab_1>python 1_1.
Enter a number: 0
Iterative: Factorial of 0 is 1
Recursive: Factorial of 0 is 1
```

Submission Requirements

1. Generate code for each task with comments.
2. Screenshots of Copilot suggestions.
3. Comparative analysis reports (Task 4 and Task 5).
4. Sample inputs/outputs demonstrating correctness.

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

	Evaluation Criteria: Criteria Max Marks Successful Setup of Copilot 0.5 Comparative Analysis – With vs Without Functions 1 Iterative vs Recursive Factorial 1 Total 2.5 Marks	

CursorAI

LabObjectives:

- ToexploreandevaluatethefunctionalityofGoogleGeminifor AI-assisted coding within Google Colab.
- TounderstandanduseCursorAIforcodegeneration, explanation, and refactoring.
- TocompareoutputsandusabilitybetweenGemini,GitHub Copilot, and Cursor AI.
- ToperformcodeoptimizationanddocumentationusingAI tools.

LabOutcomes (LOs):

Aftercompletingthislab,studentswillbeable to:

- GeneratePythoncodeusingGoogleGemini inGoogleColab.
- Analyzetheeffectivenessofcodeexplanationsandsuggestions by Gemini.
- SetupanduseCursorAI forAI-poweredcoding assistance.
- EvaluateandrefactorcodeusingCursorAIfeatures.
- CompareAItoolbehaviorandcodequalityacrossdifferent platforms.

TaskDescription#1

- Use Google Gemini in Colab to write a Python function that readsalistofnumbersandcalculatesthemean,minimum,and maximum values.

ExpectedOutput#1

- Functionalcodewithcorrectoutputandscreenshot.

Task Description #1 ● Use Google Gemini in Colab to write a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values. Expected Output #1 ● Functional code with correct output and screenshot.

```
Start coding or generate with AI.
```

```
def analyze_numbers(numbers):
  """
  Calculates the mean, minimum, and maximum values from a list of numbers.

  Args:
    numbers: A list of numbers (integers or floats).

  Returns:
    A dictionary containing the mean, minimum, and maximum values.
    Returns None if the input list is empty.
  """
  if not numbers:
    return None

  mean_value = sum(numbers) / len(numbers)
  min_value = min(numbers)
  max_value = max(numbers)

  return {
      "mean": mean_value,
      "minimum": min_value,
      "maximum": max_value
  }
```