

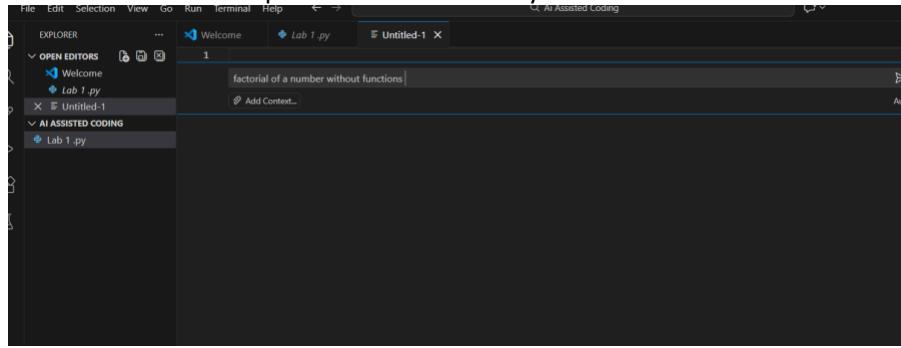
Assignment-1.2

S.Aishwarya
2303A51833 - Batch-26

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	
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. Rapelly Nandini 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	Week1 - Tuesday	Time(s)	23CSBTB01 To 23CSBTB52
Duration	2 Hours	Applicable to Batches	All batches
Assignment Number: 1.2(Present assignment number)/ 24 (Total number of assignments)			
Q.No.	Question		Expected Time to complete

1	<p>Lab 1: Environment Setup – <i>Github Copilot and VS Code Integration + Understanding AI-assisted Coding Workflow</i></p> <p>Lab Objectives:</p> <ul style="list-style-type: none"> • To install and configure GitHub Copilot in Visual Studio Code. • To explore AI-assisted code generation using GitHub Copilot. • To analyze the accuracy and effectiveness of Copilot's code suggestions. • To understand prompt-based programming using comments and code context <p>Lab Outcomes (LOs):</p> <p>After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> • 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. 	Week1 - Monday
	<p>Task 0</p> <ul style="list-style-type: none"> • Install and configure GitHub Copilot in VS Code. Take screenshots of each step. <p>Expected Output</p> <ul style="list-style-type: none"> • Install and configure GitHub Copilot in VS Code. Take screenshots of each step. 	
	<p>Task 1: AI-Generated Logic Without Modularization (Factorial without Functions)</p> <ul style="list-style-type: none"> • Scenario <p>You are building a small command-line utility for a startup intern onboarding task. The program is simple and must be written quickly without modular design.</p> <ul style="list-style-type: none"> • Task Description <p>Use GitHub Copilot to generate a Python program that computes a mathematical product-based value (factorial-like logic) directly in the main execution flow, without using any user-defined functions.</p> <ul style="list-style-type: none"> • Constraint: <ul style="list-style-type: none"> ➤ Do not define any custom function ➤ Logic must be implemented using loops and variables only 	

- **Expected Deliverables**
- A working Python program generated with Copilot assistance
- Screenshot(s) showing:
- The prompt you typed
- Copilot's suggestions
- Sample input/output screenshots
- Brief reflection (5–6 lines):
- How helpful was Copilot for a beginner?
- Did it follow best practices automatically?



```

1 # Factorial calculation without using functions
2 # Direct implementation in main execution flow
3
4 number = int(input("Enter a number to calculate its factorial: "))
5
6 if number < 0:
7     print("Factorial is not defined for negative numbers.")
8 elif number == 0 or number == 1:
9     print(f"Factorial of {number} is 1")
10 else:
11     result = 1
12     for i in range(2, number + 1):
13         result *= i
14     print(f"Factorial of {number} is {result}")
15
16

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\AI Assisted Coding> & C:/Users/aishw/AppData/Local/Programs/Python/Python312/python.exe "C:/AI Assisted Coding/Lab 1
Enter a number to calculate its factorial: 12
Factorial of 12 is 479001600
PS C:\AI Assisted Coding>

```

Explanation: This Program Calculates the factorial of a number.If negative number is entered the factorial is not defined .

Task 2: AI Code Optimization & Cleanup (Improving Efficiency)

❖ Scenario

Your team lead asks you to **review AI-generated code** before committing it to a shared repository.

❖ Task Description

Analyze the code generated in **Task 1** and use Copilot again to:

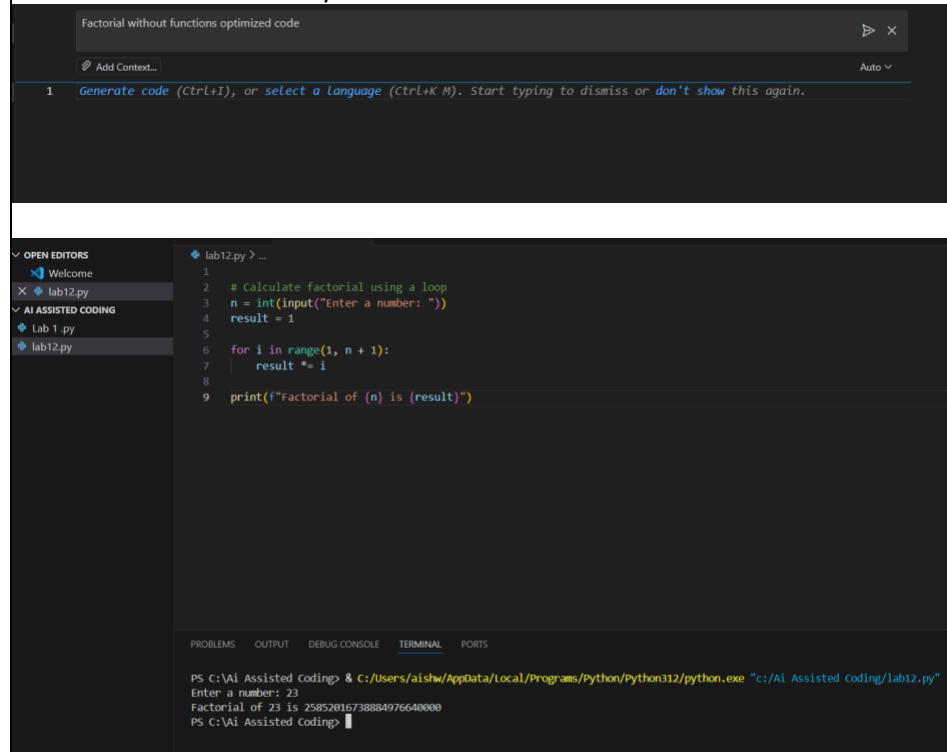
- Reduce unnecessary variables
- Improve loop clarity
- Enhance readability and efficiency

Hint:

Prompt Copilot with phrases like
"optimize this code", *"simplify logic"*, or *"make it more readable"*

❖ **Expected Deliverables**

- Original AI-generated code
- Optimized version of the same code
- Side-by-side comparison
- Written explanation:
 - What was improved?
 - Why the new version is better (readability, performance, maintainability).



The screenshot shows a code editor interface with the following details:

- Title Bar:** Factorial without functions optimized code
- Toolbar:** Add Context... (dropdown), Auto
- Message Bar:** 1 Generate code (Ctrl+I), or select a Language (Ctrl+K M). Start typing to dismiss or don't show this again.
- Sidebar:** OPEN EDITORS (Welcome, lab12.py, Lab 1.py, lab12.py) and AI ASSISTED CODING (Lab 1.py, lab12.py).
- Code Editor:**

```

1 # Calculate Factorial using a loop
2 n = int(input("Enter a number: "))
3 result = 1
4 for i in range(1, n + 1):
5     result *= i
6
7 print(f"Factorial of {n} is {result}")

```
- Terminal:**

```

PS C:\Ai Assisted Coding> & C:/Users/aishw/AppData/Local/Programs/Python/Python312/python.exe "c:/Ai Assisted Coding/lab12.py"
Enter a number: 23
Factorial of 23 is 25852016738884976640000
PS C:\Ai Assisted Coding>

```
- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS

Explain : Optimized code removes unnecessary conditions simplifies the logic which makes it more understandable and maintainable .

Task 3: Modular Design Using AI Assistance (Factorial with Functions)

❖ **Scenario**

The same logic now needs to be reused in **multiple scripts**.

❖ **Task Description**

Use GitHub Copilot to generate a **modular version** of the program by:

- Creating a **user-defined function**
- Calling the function from the main block

❖ **Constraints**

- Use meaningful function and variable names
- Include inline comments (preferably suggested by Copilot)

❖ **Expected Deliverables**

- AI-assisted function-based program
- Screenshots showing:
 - Prompt evolution
 - Copilot-generated function logic
- Sample inputs/outputs
- Short note:
 - How modularity improves reusability.

Factorial with Functions

Add Context...

10

File Explorer

... Welcome lab12.py

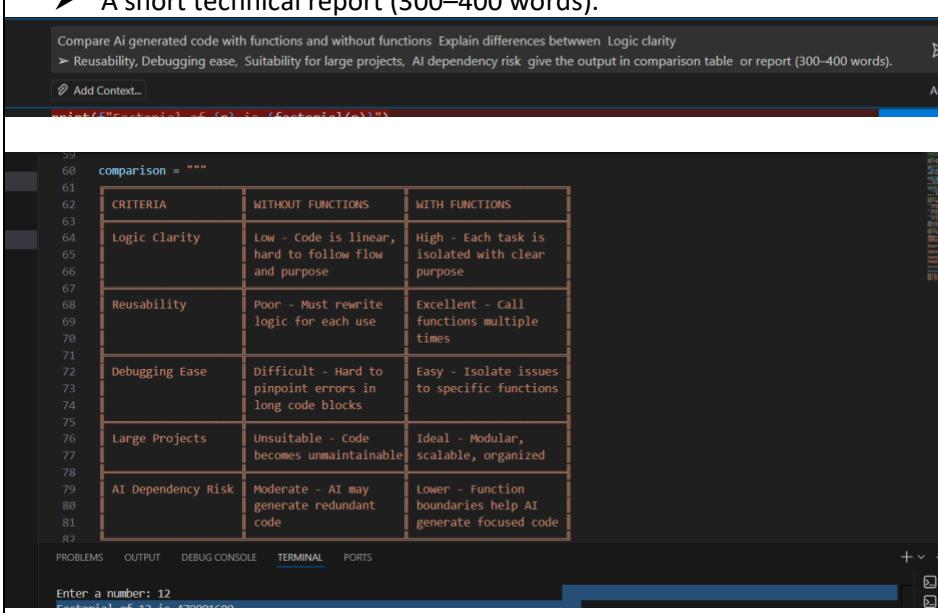
```

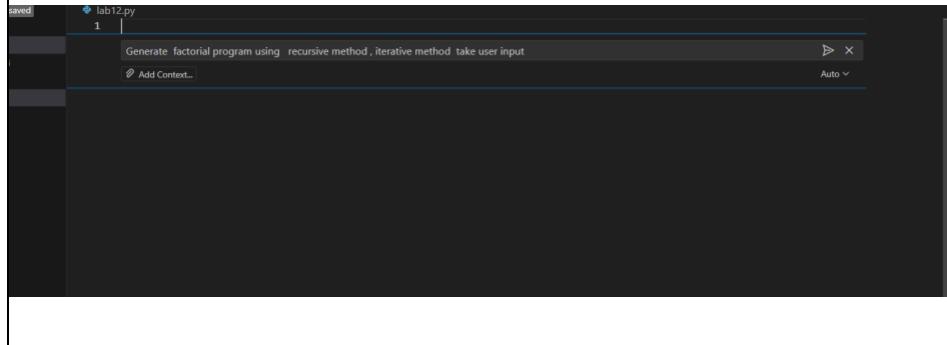
DRS
ome
-py
D CODING
/
1 # calculate factorial using a function
2 def factorial(num):
3     if num < 0:
4         return "Error: negative number"
5     elif num == 0 or num == 1:
6         return 1
7     else:
8         result = 1
9         for i in range(2, num + 1):
10            result *= i
11     return result
12
13
14 n = int(input("Enter a number: "))
15 print(f"Factorial of {n} is {factorial(n)}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Enter a number: 12
Factorial of 12 is 479001600
Enter a number: 24
Factorial of 24 is 620448401733239439360000
PS C:\AI Assisted coding> ^C
PS C:\AI Assisted coding> c: cd "c:\AI Assisted coding"; & "c:\Users\Aishw\appData\local\Programs\Python\Python312\python.exe" "c:\Users\Aishw\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher" "52922" ... "c:\AI Assisted coding\lab12.py"
Enter a number: 23
Factorial of 23 is 25852016738884976640000
PS C:\AI Assisted coding>

	<p>Explanation : This Program calculates the factorial using user defined function . This condition avoids unnecessary calculations .</p> <p>Task 4: Comparative Analysis – Procedural vs Modular AI Code (With vs Without Functions)</p> <p>❖ Scenario As part of a code review meeting, you are asked to justify design choices.</p> <p>❖ Task Description Compare the non-function and function-based Copilot-generated programs on the following criteria:</p> <ul style="list-style-type: none"> ➤ Logic clarity ➤ Reusability ➤ Debugging ease ➤ Suitability for large projects ➤ AI dependency risk <p>❖ Expected Deliverables Choose one:</p> <ul style="list-style-type: none"> ➤ A comparison table OR ➤ A short technical report (300–400 words).  <table border="1" data-bbox="448 1362 922 1721"> <thead> <tr> <th>CRITERIA</th> <th>WITHOUT FUNCTIONS</th> <th>WITH FUNCTIONS</th> </tr> </thead> <tbody> <tr> <td>Logic Clarity</td> <td>Low - Code is linear, hard to follow flow and purpose</td> <td>High - Each task is isolated with clear purpose</td> </tr> <tr> <td>Reusability</td> <td>Poor - Must rewrite logic for each use</td> <td>Excellent - Call functions multiple times</td> </tr> <tr> <td>Debugging Ease</td> <td>Difficult - Hard to pinpoint errors in long code blocks</td> <td>Easy - Isolate issues to specific functions</td> </tr> <tr> <td>Large Projects</td> <td>Unsuitable - Code becomes unmaintainable</td> <td>Ideal - Modular, scalable, organized</td> </tr> <tr> <td>AI Dependency Risk</td> <td>Moderate - AI may generate redundant code</td> <td>Lower - Function boundaries help AI generate focused code</td> </tr> </tbody> </table> <p>Explanation : This table shows that writing code with functions is better than writing code without functions because functions make the logic clearer, allow</p>	CRITERIA	WITHOUT FUNCTIONS	WITH FUNCTIONS	Logic Clarity	Low - Code is linear, hard to follow flow and purpose	High - Each task is isolated with clear purpose	Reusability	Poor - Must rewrite logic for each use	Excellent - Call functions multiple times	Debugging Ease	Difficult - Hard to pinpoint errors in long code blocks	Easy - Isolate issues to specific functions	Large Projects	Unsuitable - Code becomes unmaintainable	Ideal - Modular, scalable, organized	AI Dependency Risk	Moderate - AI may generate redundant code	Lower - Function boundaries help AI generate focused code	
CRITERIA	WITHOUT FUNCTIONS	WITH FUNCTIONS																		
Logic Clarity	Low - Code is linear, hard to follow flow and purpose	High - Each task is isolated with clear purpose																		
Reusability	Poor - Must rewrite logic for each use	Excellent - Call functions multiple times																		
Debugging Ease	Difficult - Hard to pinpoint errors in long code blocks	Easy - Isolate issues to specific functions																		
Large Projects	Unsuitable - Code becomes unmaintainable	Ideal - Modular, scalable, organized																		
AI Dependency Risk	Moderate - AI may generate redundant code	Lower - Function boundaries help AI generate focused code																		

	<p>code reuse, simplify debugging, and help manage large projects efficiently. Code without functions is harder to understand, difficult to maintain, and often leads to repetitive or messy code, while functions keep programs modular and organized.</p>	
	<p>Task 5: AI-Generated Iterative vs Recursive Thinking</p> <ul style="list-style-type: none"> ❖ Scenario Your mentor wants to test how well AI understands different computational paradigms. ❖ Task Description Prompt Copilot to generate: An iterative version of the logic A recursive version of the same logic ❖ Constraints Both implementations must produce identical outputs Students must not manually write the code first ❖ Expected Deliverables Two AI-generated implementations Execution flow explanation (in your own words) Comparison covering: <ul style="list-style-type: none"> ➢ Readability ➢ Stack usage ➢ Performance implications ➢ When recursion is <i>not</i> recommended. 	

```
ITORS
lcome
l2.py
py
py

Welcome > lab12.py > ...
1 def factorial_recursive(n):
2     """Calculate factorial using recursion"""
3     if n < 0:
4         return "Error: Factorial not defined for negative numbers"
5     elif n == 0 or n == 1:
6         return 1
7     else:
8         return n * factorial_recursive(n - 1)
9
10
11 def factorial_iterative(n):
12     """Calculate factorial using iteration"""
13     if n < 0:
14         return "Error: Factorial not defined for negative numbers"
15
16     result = 1
17     for i in range(2, n + 1):
18         result *= i
19     return result
20
21
22 # Main program
23 if __name__ == "__main__":
24     try:
25         num = int(input("Enter a number to calculate factorial: "))
26
27         print(f"\nUsing Recursive Method: {factorial_recursive(num)}")
28         print(f"Using Iterative Method: {factorial_iterative(num)}")
29     except ValueError:
30         print("Error: Please enter a valid integer")
```

PS C:\Ai Assisted Coding> & C:/Users/dishw/AppData/Local/Programs/Python/Python311/python.exe lab12.py
Enter a number to calculate factorial: 23

Using Recursive Method: 25852016738884976640000
Using Iterative Method: 25852016738884976640000
PS C:\Ai Assisted Coding>

Explanation : This program finds the factorial of a number using both recursive and iterative methods. The recursive function calls itself until it reaches the base case, while the iterative function uses a loop to calculate the result. Both methods give the same output for valid inputs. Recursion is easier to understand but uses more memory, whereas iteration is more efficient and safer for large numbers.

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 as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.