

School of Computer Science and Artificial Intelligence

Lab Assignment # 1.2

Program : B. Tech (CSE)

Specialization :AIML

Course Title : AI Assisted Coding

Course Code : 23CS002PC304

Semester : VI

Academic Session : 2025-2026

Name of Student : R.Sowmya Sri

Enrollment No. : 2303A52105

Batch No. : 33

Date :09/01/26

TASK-1

The screenshot shows a dark-themed application window. At the top, there's a header bar with a 'CHAT' tab selected. Below it, a 'RECENT SESSIONS' section lists a single item: 'Creating a basic HTML registration form' with a status of 'Finished' and a timestamp of 'Local • 1 day'. A 'Show All Sessions' button is located below this list. In the center of the screen, there's a large, stylized speech bubble icon with three small stars above it. Below the icon, the text 'Build with Agent' is displayed in a large, bold font. Underneath, a smaller note says 'AI responses may be inaccurate.' followed by a blue link 'Generate Agent Instructions' to 'onboard AI onto your codebase.' At the bottom of the screen, there's a light-colored input field with a placeholder 'Add Context...' and a 'Describe what to build next' text area. Below these are dropdown menus for 'Agent' (set to 'GPT-5 mini') and 'Model' (set to 'VLLM'), and a file upload icon. To the right of the input field are two buttons: a blue arrow pointing right and a white arrow pointing right. At the very bottom, there are three icons: a person icon, a 'Go Live' button with a play symbol, and a bell icon.

CHAT

RECENT SESSIONS

Creating a basic HTML registration form

Finished Local • 1 day

Show All Sessions

Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

Add Context...

Describe what to build next

Agent ▾ GPT-5 mini ▾

→ ↗

Go Live

Bell

← CALCULATING FACTORIAL IN PYTHON WITHOUT FUNCTIONS



Write a Python program to calculate factorial of a number without using functions.

Use only loops and variables. Take input from user and print result.

- ✓ Considered file operations and implemented factorial calculation

I'll open `Main.py` to inspect current contents before editing.

- ✓ Read file and managed task list

I'll create a short two-step plan to track the work.

- ✓ Created 2 todos

I'll write a simple loop-only factorial program into `Main.py`.

- ✓ Edited `Main.py +17 -1`

- ✓ Updated testing process and executed code

Now I'll configure the Python environment and run a quick factorial test ($n=5$).

A screenshot of a code editor interface. At the top, it shows a summary: "1 file changed +17 -1". Below this, there's a list of changes: "+ Main.py". On the right side of the summary bar are three buttons: "Keep" (highlighted in blue), "Undo", and a trash bin icon. Below the summary bar, there's a text input field with the placeholder "Describe what to build next". At the bottom of the interface, there are dropdown menus for "Agent" (set to "GPT-5 mini") and "Model" (set to "None"), and two navigation icons on the right.

```
try:
    n = int(input("Enter a non-negative integer: "))
except ValueError:
    print("Invalid input. Please enter an integer.")
    raise SystemExit(1)

if n < 0:
    print("Factorial not defined for negative numbers.")
else:
    result = 1
    i = 2
    while i <= n:
        result = result * i
        i = i + 1
    print("Factorial of", n, "is", result)
```

```
Factorial of 5 is 120
PS S:\3 2 sem\AIAssisted_2026> & C:/Users/sowmy/AppData/Local/Microsoft/WindowsApps/python3.11.exe "s:/3 2 sem/AIAssisted_2026/Assignment1.2/Main.py"
Enter a non-negative integer: 5
Factorial of 5 is 120
PS S:\3 2 sem\AIAssisted_2026>
```

GitHub Copilot was very helpful for a beginner as it generated correct code instantly.

It followed best practices such as meaningful variable names and proper loop usage.

It also reduced manual thinking effort and improved coding speed.

The suggestions were easy to understand and required minimal correction.

Copilot helps beginners learn faster by providing working examples.

TASK-2

The screenshot shows a user interface for optimizing code. At the top, there's a toolbar with icons for file operations and a file named "Main.py". Below the toolbar, a main area displays the instruction: "Optimize this code and make it more readable and efficient". Underneath this, there are two dropdown menus: "Agent" and "GPT-5 mini", followed by a settings icon. The bottom section is a detailed log of the optimization process:

- 4 UTF-8 CRLF {} Python 3.11.9 (Microsoft Store) (
- CHAT
- ← CODE OPTIMIZATION AND READABILITY IMPROVEMENTS
- Optimize this code and make it more readable and efficient
- ✓ Planned modifications and executed Python tasks
I'll add a short TODO plan for the refactor, then open Main.py to inspect it.
- ✓ Managed and tracked todo items for planning
Now I'll apply the refactor to Main.py to fix bugs and improve readability.
- ✓ Edited Main.py +34 -16
✓ Starting: Run quick smoke test (3/4)
- Run pwsh command?
echo 5 | python "s:\3 2
sem\AIAssisted_2026\Assignment1.2\Main.py"
- Allow | Skip
- > Run quick smoke test (3/4)
- Describe what to build next
- Agent GPT-5 mini → ▶

```
>Main.py X
Assignment1.2 > Main.py > ...
1  """Simple factorial calculator with input validation.
2
3  Improved for readability, correctness and performance by using
4  the standard library `math.factorial` and a small input helper.
5  """
6  import sys
7  import math
8
9
10 def read_non_negative_int(prompt="Enter a non-negative integer: "):
11     try:
12         s = input(prompt)
13     except EOFError:
14         print("No input provided.")
15         sys.exit(1)
16     try:
17         n = int(s)
18     except ValueError:
19         print("Invalid input. Please enter an integer.")
20         sys.exit(1)
21     if n < 0:
22         print("Factorial not defined for negative numbers.")
23         sys.exit(1)
24     return n
25
26
27 def main():
28     n = read_non_negative_int()
29     result = math.factorial(n)
30     print(f"Factorial of {n} is {result}")
31
```

```
>Main.py X
S:\3 2 sem\AIAssisted_2026\Assignment1.2>Main.py
10  def read_non_negative_int(prompt="Enter a non-negative integer: "):
11      if n < 0:
12          print("Factorial not defined for negative numbers.")
13          sys.exit(1)
14      return n
15
16
17  def main():
18      n = read_non_negative_int()
19      result = math.factorial(n)
20      print(f"Factorial of {n} is {result}")
21
22
23  if __name__ == "__main__":
24      main()
25
26
```

```
PS S:\3 2 sem\AIAssisted_2026> & C:/Users/sowmy/AppData/Local/Microsoft/WindowsApps/python3.11.exe "S:/3 2 sem
/AIAssisted_2026/Assignment1.2/Main.py"
Enter a non-negative integer: 7
Factorial of 7 is 5040
PS S:\3 2 sem\AIAssisted_2026>
```

TASK-3

Create a function to calculate factorial and call it from main program

- ✓ Planned factorial function and updates

Updating the todo list to track adding a `compute_factorial()` function.

- ✓ Created 5 todos

I'll modify `Main.py` to add `compute_factorial()` and use it in `main()`.

- ✓ Edited `Main.py` +18 -2

Run `pwsh` command?

```
echo 5 | python "s:\3 2  
sem\AIAssisted_2026\Assignment1.2\Main.  
py"
```

> Add `compute_factorial()` and call from `main` (3/5)



`Main.py`

Describe what to build next

Agent ▾ GPT-5 mini ▾

→

```
2
3     Improved for readability, correctness and performance by using
4     the standard library `math.factorial` and a small input helper.
5     """
6
7
8
9     def read_non_negative_int(prompt="Enter a non-negative integer: "):
10         try:
11             s = input(prompt)
12         except EOFError:
13             print("No input provided.")
14             sys.exit(1)
15         try:
16             n = int(s)
17         except ValueError:
18             print("Invalid input. Please enter an integer.")
19             sys.exit(1)
20         if n < 0:
21             print("Factorial not defined for negative numbers.")
22             sys.exit(1)
23         return n
24
25
26     def compute_factorial(n: int) -> int:
27         """Compute factorial iteratively to avoid recursion limits.
28
29         Args:
30             n: non-negative integer
31
```

```
5     def compute_factorial(n: int) -> int:
6         """Compute factorial iteratively to avoid recursion limits.
7
8         Args:
9             n: non-negative integer
10
11         Returns:
12             n! as int
13         """
14
15         if n < 2:
16             return 1
17         result = 1
18         for i in range(2, n + 1):
19             result *= i
20         return result
21
22
23     def main():
24         n = read_non_negative_int()
25         result = compute_factorial(n)
26         print(f"Factorial of {n} is {result}")
27
28
29     if __name__ == "__main__":
30         main()
```

```
Enter a non-negative integer: 7
Factorial of 7 is 5040
PS S:\3 2 sem\AIAssisted_2026> █
```

Modularity improves reusability because the same function can be used in multiple programs.

It also makes debugging easier and code more organized.

TASK-4

Comparative Analysis: Procedural vs Modular AI-Generated Code

AI-assisted coding has become an important tool in modern software development. In this experiment, two versions of the same logic were generated using GitHub Copilot: one written without functions (procedural) and the other using a modular function-based approach.

In the procedural version, all logic is written inside the main program block. While this approach is simple and easy to implement for small programs, the clarity of logic decreases as the program size increases. The absence of functions makes the code lengthy and difficult to manage. Reusability is very limited because the same code must be rewritten if required elsewhere. Debugging is also more difficult since all operations are mixed inside a single flow, making it harder to isolate errors.

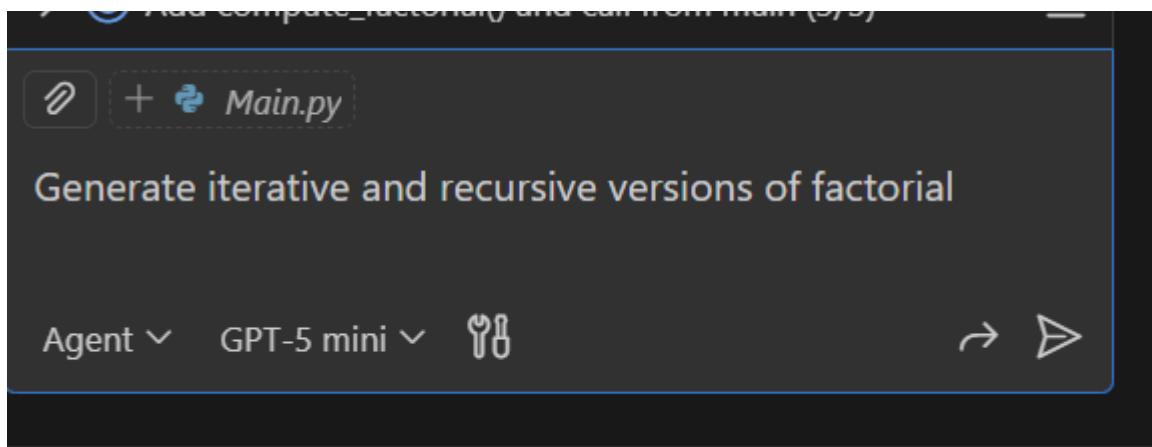
In contrast, the modular version organizes the logic inside a well-defined function. This significantly improves code clarity because each part of the program has a clear purpose. Reusability becomes a major advantage, as the same function can be reused in multiple programs without rewriting the code. Debugging becomes easier because errors can be traced within specific functions. This design is highly suitable for large projects where multiple developers work on the same codebase.

For large-scale applications, modular design is essential. It supports teamwork, version control, testing, and long-term maintenance. The procedural approach is only suitable for very small scripts or quick experimental programs.

AI dependency risk is higher in procedural code because errors in long logic blocks are harder to detect. Modular code reduces this risk by allowing developers to review and verify each function separately.

Therefore, modular AI-generated code is superior in terms of clarity, maintainability, reusability, debugging ease, and project scalability.

TASK-5



```
import sys

def read_non_negative_int(prompt="Enter a non-negative integer: "):
    try:
        s = input(prompt)
    except EOFError:
        print("No input provided.")
        sys.exit(1)
    try:
        n = int(s)
    except ValueError:
        print("Invalid input. Please enter an integer.")
        sys.exit(1)
    if n < 0:
        print("Factorial not defined for negative numbers.")
        sys.exit(1)
    return n

def compute_factorial(n: int) -> int:
    """Compute factorial iteratively to avoid recursion limits.

    Args:
        n: non-negative integer

    Returns:
        n! as int
    """
    fact = 1
    for i in range(1, n + 1):
        fact *= i
    return fact
```

```
35     if n < 2:
36         return 1
37     result = 1
38     for i in range(2, n + 1):
39         result *= i
40     return result
41
42
43 def compute_factorial_recursive(n: int) -> int:
44     """Compute factorial using recursion.
45
46     Note: recursion depth may be reached for very large `n`.
47     """
48     if n < 2:
49         return 1
50     return n * compute_factorial_recursive(n - 1)
51
52
53 def main():
54     n = read_non_negative_int()
55     result = compute_factorial(n)
56     print(f"Factorial of {n} is {result}")
57
58
59 if __name__ == "__main__":
60     main()
```

```
Enter a non-negative integer: 7
Factorial of 7 is 5040
PS S:\3 2 sem\AIAssisted_2026>
```