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

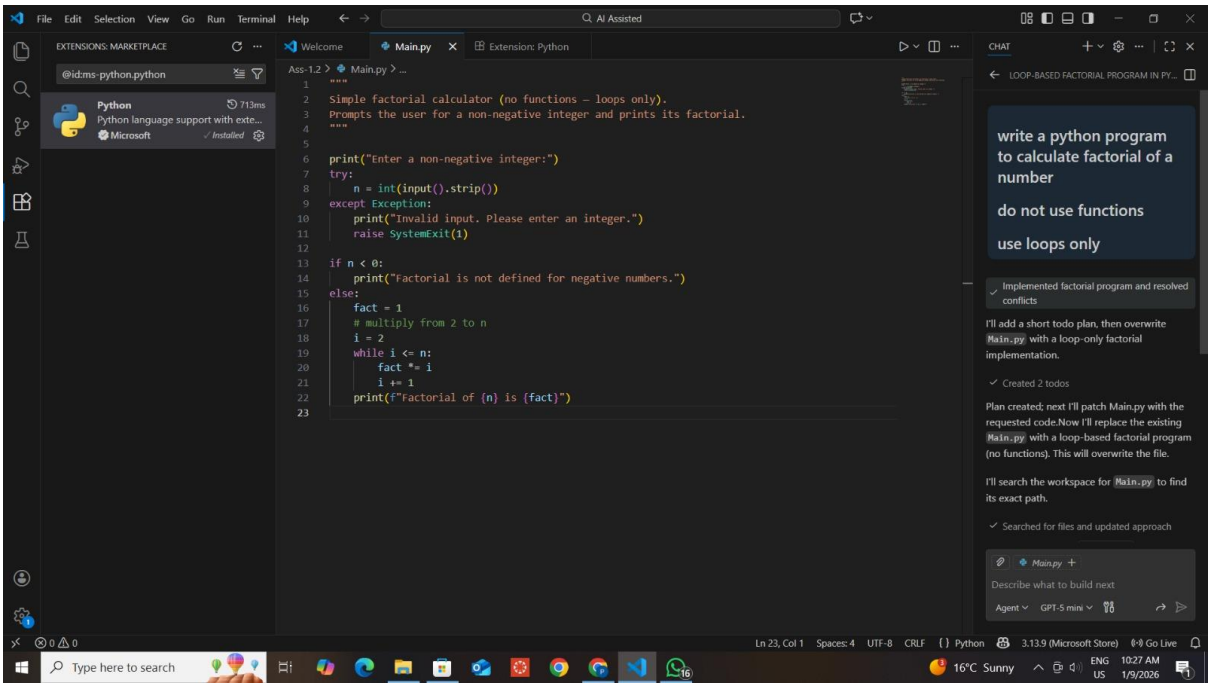
Name of Student :P.Sushanthi

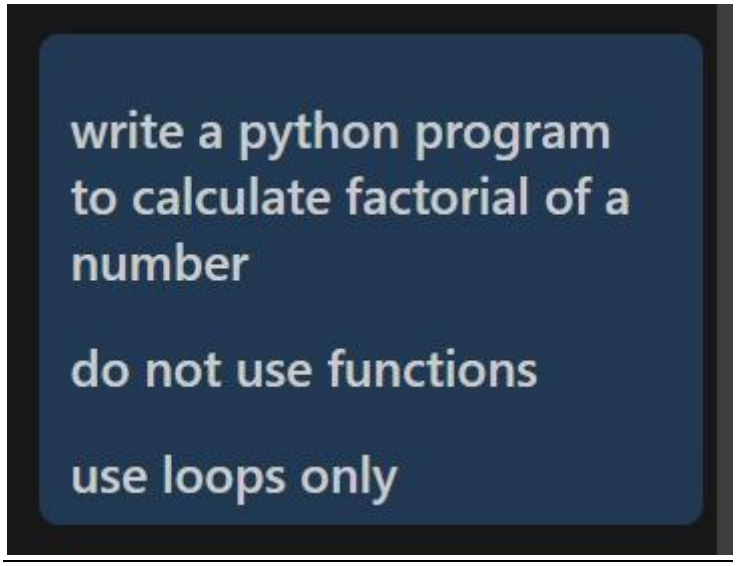
Enrollment No : 2303A52102

Batch No : 33

Date :09/01/26

TASK-1



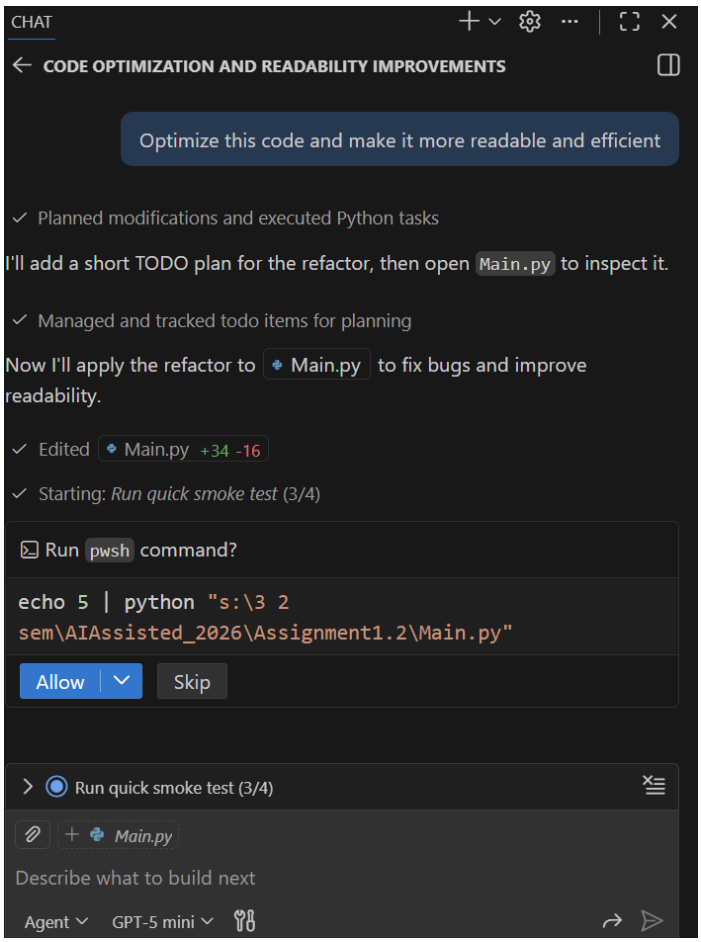


GitHub Copilot was very helpful for a beginner as it generated correct code instantly.

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



Main.py

Assignment1.2 > Main.py > ...

```python
"""Simple factorial calculator with input validation.

Improved for readability, correctness and performance by using
the standard library `math.factorial` and a small input helper.
"""
import sys
import math


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 main():
    n = read_non_negative_int()
    result = math.factorial(n)
    print(f"Factorial of {n} is {result}")

```

Main.py

S:\3 2 sem\AIAssisted_2026\Assignment1.2\Main.py

```python
def read_non_negative_int(prompt="Enter a non-negative integer: "):
    if n < 0:
        print("Factorial not defined for negative numbers.")
        sys.exit(1)
    return n


def main():
    n = read_non_negative_int()
    result = math.factorial(n)
    print(f"Factorial of {n} is {result}")


if __name__ == "__main__":
    main()

```

```
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

write a python program to calculate factorial of a number

do not use functions

use loops only

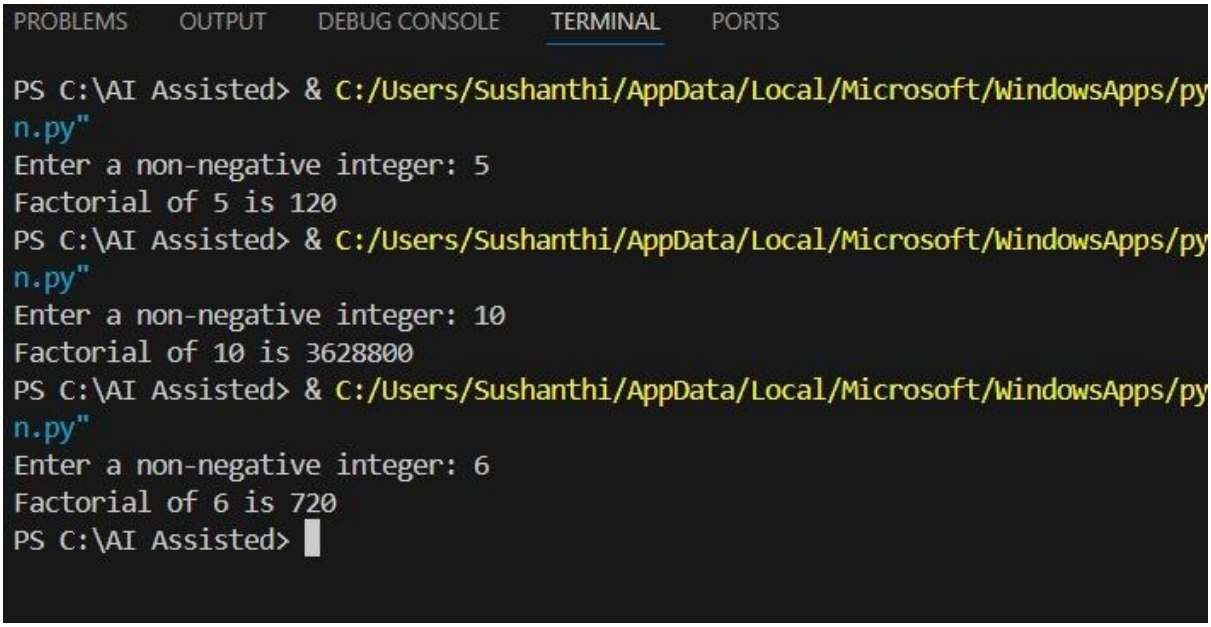
Main.py > main

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

    # Handle negative inputs explicitly
    if n < 0:
        print("Factorial is not defined for negative numbers.")
        return

    # Compute and display the factorial using the function
    print(f"Factorial of {n} is {factorial(n)}")


if __name__ == "__main__":
    main()
```



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

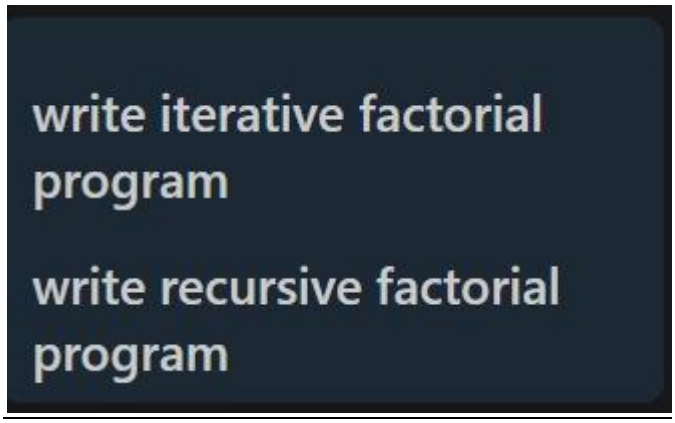
same function can be reused in multiple programs without rewriting the code. Debugging becomes easier because errors can be traced within specific functions. This design is highly suitable for large projects where multiple developers work on the same codebase.

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



```
Main.py > factorial > factorial_iterative > result
def factorial(n):
    def factorial_iterative(n):
        Returns:
            int: n!
        """
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result


    def factorial_recursive(n):
        """Recursive factorial implementation.

        Uses the relation n! = n * (n-1)! with base cases 0! = 1 and 1! = 1.
        """
        if n <= 1:
            return 1
        return n * factorial_recursive(n - 1)


    def main():
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    Python

7
PS C:\AI Assisted> & C:/Users/Sushanthi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/AI Assisted/Ass-1.2/Mai
n.py"
PS C:\AI Assisted> 7
7
PS C:\AI Assisted> & C:/Users/Sushanthi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/AI Assisted/Ass-1.2/Mai
n.py"
PS C:\AI Assisted> & C:/Users/Sushanthi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/AI Assisted/Ass-1.2/Mai
n.py"
PS C:\AI Assisted> & C:/Users/Sushanthi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/AI Assisted/Ass-1.2/Mai
n.py"
PS C:\AI Assisted> 7
7
PS C:\AI Assisted>

Ln 30, Col 14   Spaces: 4   UTF-8   CRLF   { } P
```