Lab-01

Topic:- Environment Setup – GitHub Copilot and VS Code Integration + Understanding AI-assisted Coding Workflow

Student Details:

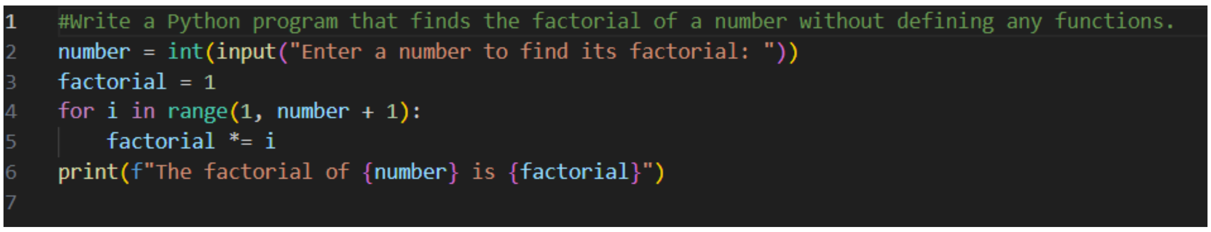
Name	: Thota Saiteja
Hall Ticket No	: 2303A51535
Batch	: 22

Task-01

Final Optimal Prompt

write a python program to calculate the factorial of a number user without using function.

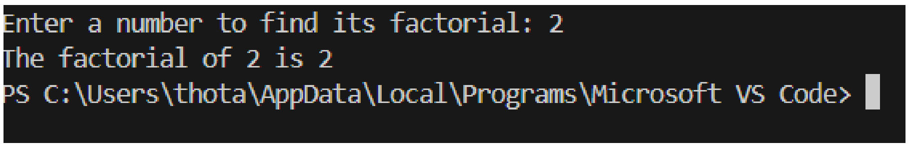
Code Screenshot



```python
#Write a Python program that finds the factorial of a number without defining any functions.
number = int(input("Enter a number to find its factorial: "))
factorial = 1
for i in range(1, number + 1):
    factorial *= i
print(f"The factorial of {number} is {factorial}")

```

Output Screenshot



```
Enter a number to find its factorial: 2
The factorial of 2 is 2
PS C:\Users\thota\AppData\Local\Programs\Microsoft VS Code>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

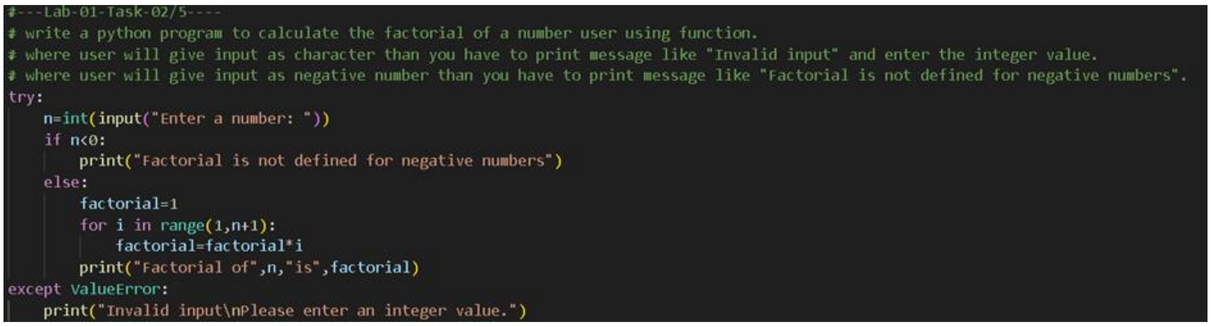
While writing the prompt for calculating the factorial without using a function in VS Code, GitHub Copilot assisted by predicting the remaining part of the prompt even before it was fully typed. This shows how AI-assisted coding improves speed and reduces manual effort. Copilot suggested a loop-based solution that follows basic Python syntax and logic. The program takes user input, validates it, and calculates the factorial using an iterative approach. This task helped in understanding how AI tools support beginners by providing instant code suggestions. It also demonstrated how Copilot understands the programmer's intent from minimal input.

Task-02

Final Optimal Prompt

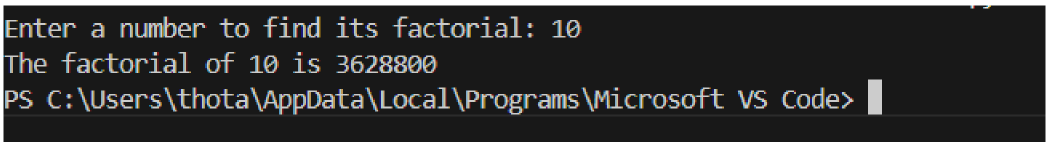
write a python program to calculate the factorial of a number user using function.
where user will give input as character than you have to print message like "Invalid input" and enter the integer value.
where user will give input as negative number than you have to print message like "Factorial is not defined for negative numbers".

Code Screenshot



```python
#---Lab-01-Task-02/5----
# write a python program to calculate the factorial of a number user using function.
# where user will give input as character than you have to print message like "Invalid input" and enter the integer value.
# where user will give input as negative number than you have to print message like "Factorial is not defined for negative numbers".
try:
    n=int(input("Enter a number: "))
    if n<0:
        print("Factorial is not defined for negative numbers")
    else:
        factorial=1
        for i in range(1,n+1):
            factorial=factorial*i
        print("Factorial of",n,"is",factorial)
except ValueError:
    print("Invalid input\nPlease enter an integer value.")
```

Output Screenshot



```
Enter a number to find its factorial: 10
The factorial of 10 is 3628800
PS C:\Users\thota\AppData\Local\Programs\Microsoft VS Code>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

In this task, GitHub Copilot assisted in writing a factorial program using a function along with input validation. When the prompt included conditions for invalid character input and negative numbers, Copilot automatically suggested appropriate conditional checks. The program displays meaningful error messages when the user enters invalid or negative input. This improves program robustness and user experience. The use of a function makes the code modular and reusable. Through this task, I observed that AI tools can generate structured and readable code when given clear and specific prompts.

Task-03

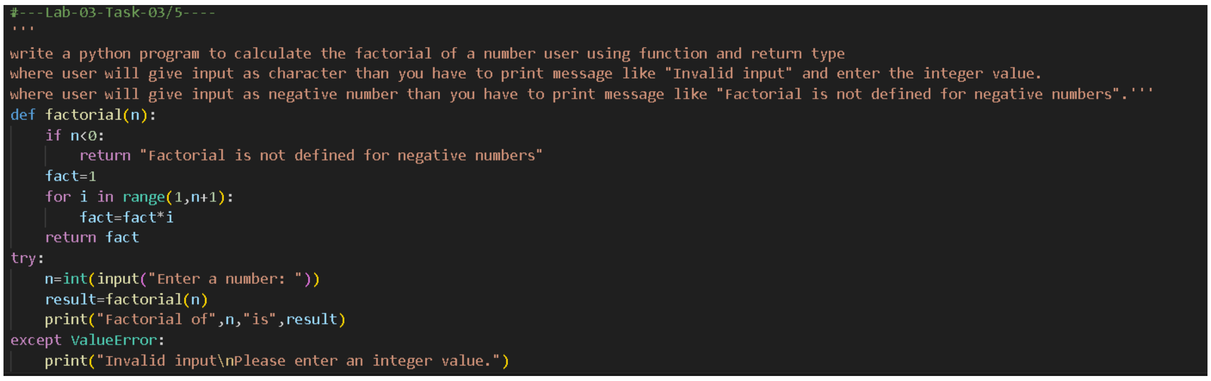
Final Optimal Prompt

write a python program to calculate the factorial of a number user using function and return type

where user will give input as character than you have to print message like "Invalid input" and enter the integer value.

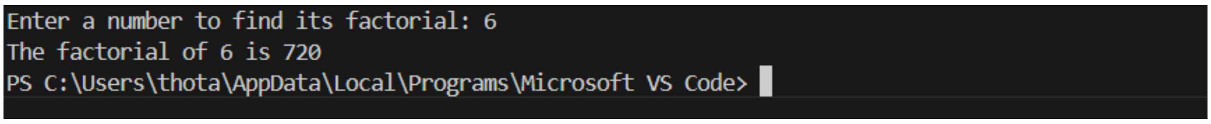
where user will give input as negative number than you have to print message like "Factorial is not defined for negative numbers".

Code Screenshot



```python
#---Lab-03-Task-03/5----
'''
write a python program to calculate the factorial of a number user using function and return type
where user will give input as character than you have to print message like "Invalid input" and enter the integer value.
where user will give input as negative number than you have to print message like "Factorial is not defined for negative numbers".'''
def factorial(n):
    if n<0:
        return "Factorial is not defined for negative numbers"
    fact=1
    for i in range(1,n+1):
        fact=fact*i
    return fact
try:
    n=int(input("Enter a number: "))
    result=factorial(n)
    print("Factorial of",n,"is",result)
except ValueError:
    print("Invalid input\nPlease enter an integer value.")
```

Output Screenshot



```
Enter a number to find its factorial: 6
The factorial of 6 is 720
PS C:\Users\thota\AppData\Local\Programs\Microsoft VS Code>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

This task extends the factorial program by using a function with a return type. GitHub Copilot efficiently generated code that returns the factorial value instead of directly printing it. The program properly handles invalid inputs such as characters and negative numbers by displaying suitable messages. Using return statements improves code flexibility and allows reuse of the result in other parts of the program. This task highlighted how AI understands programming concepts like return values and input validation. It also showed that detailed prompts result in more accurate and optimized code suggestions.

Task-04

Criteria	Procedural Code (Without Functions)	Modular Code (With Functions)
Logic Clarity	The factorial logic is written directly in the main program along with input and output statements. This makes the flow harder to understand as the program grows.	The factorial calculation is placed inside a dedicated function, making the logic clear, structured, and easy to read.
Reusability	The factorial logic cannot be reused without copying the same code again, leading to repetition.	The function can be reused multiple times by simply calling it, improving code reuse.
Debugging Ease	Debugging is difficult because all logic exists in one block, making it harder to isolate errors.	Debugging is easier since errors can be traced directly to the factorial function.
Suitability for Large Projects	Not suitable for large projects, as the code becomes lengthy and difficult to maintain.	Suitable for large projects because modular code is easier to manage, extend, and maintain.
AI Dependency Risk	Higher dependency on AI tools to rewrite or fix code due to poor structure.	Lower dependency on AI once the function logic is understood and maintained manually.

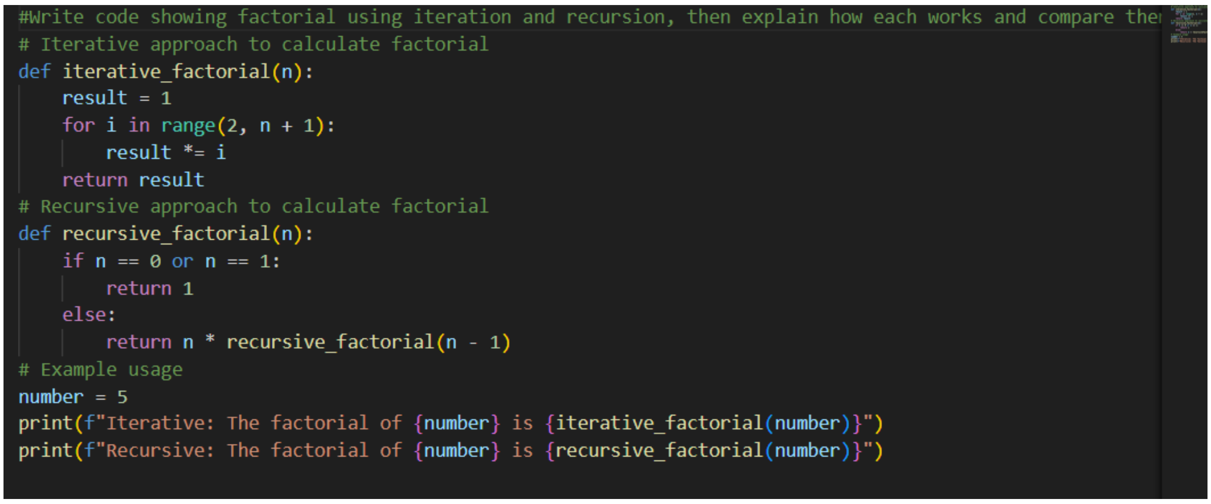
Although a factorial program is simple, using a **function-based (modular) approach** is more effective. It improves clarity, reusability, and maintainability while reducing long-term dependency on AI tools. Therefore, modular design is the preferred approach for both academic and real-world software development.

Tack-05

Final Optimal Prompt

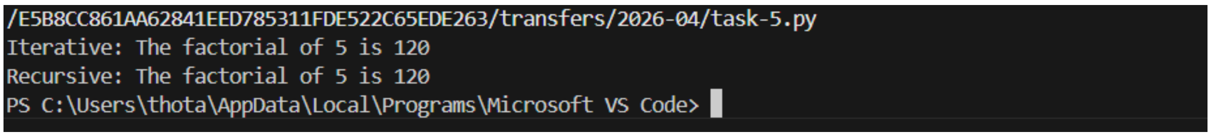
write a python program to calculate the factorial of a number user using recursive function as well as iterative function. where user will give input as character than you have to print message like "Invalid input" and enter the integer value. where user will give input as negative number than you have to print message like "Factorial is not defined for negative numbers" and where user will give input in float than you have to print message like "Factorial is not defined for decimal numbers(float)"

Code Screenshot



```python
#Write code showing factorial using iteration and recursion, then explain how each works and compare the
# Iterative approach to calculate factorial
def iterative_factorial(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
# Recursive approach to calculate factorial
def recursive_factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * recursive_factorial(n - 1)
# Example usage
number = 5
print(f"Iterative: The factorial of {number} is {iterative_factorial(number)}")
print(f"Recursive: The factorial of {number} is {recursive_factorial(number)}")
```

Output Screenshot



```
/E5B8CC861AA62841EED785311FDE522C65EDE263/transfers/2026-04/task-5.py
Iterative: The factorial of 5 is 120
Recursive: The factorial of 5 is 120
PS C:\Users\thota\AppData\Local\Programs\Microsoft VS Code>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

This task involves calculating factorial using both iterative and recursive functions with comprehensive input validation. GitHub Copilot accurately suggested both approaches and handled conditions for character, negative, and float inputs. The program displays clear messages for each invalid case, ensuring correct user interaction. Using both methods helped compare performance and logic differences between recursion and iteration. This task showed that AI can generate complex logic when the prompt is well-defined. It also reinforced the importance of handling edge cases in real-world programming.