SR UNIVERSITY

School Of Computer Science and Artificial Intelligence

Lab Assignment – 1.2

Name : Mohammed Sabir

Enrollment No : 2303A51506

Batch No : 22

Task 1 : AI-Generated Logic Without Modularization (Factorial without Functions)

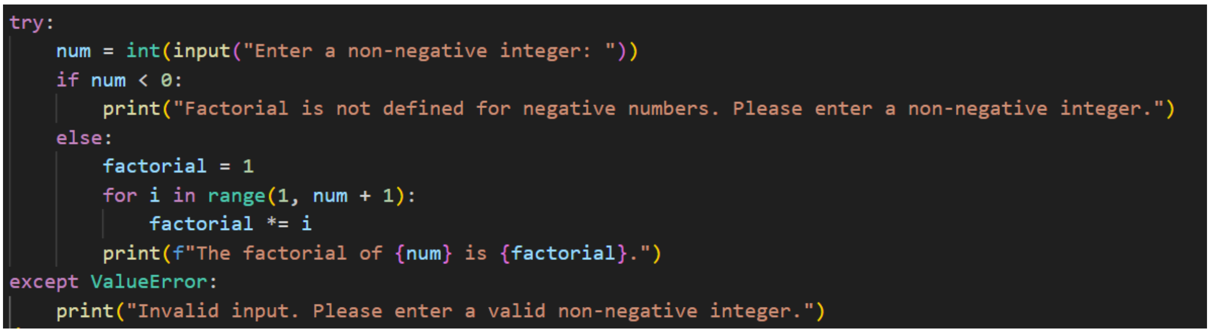
Prompt :-

#write a python code for factorial of a number without using function

#for factorial of a number without using function and tell if we type a negative number it should not calculate factorial and tell the user give a string input for these scenarios check the code below

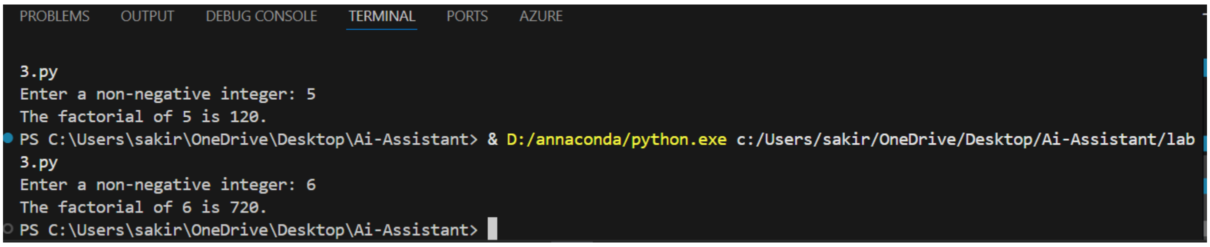
#if the user give a string input it should handle the error and tell the user to give a valid input.

Code :-



```
try:
    num = int(input("Enter a non-negative integer: "))
    if num < 0:
        print("Factorial is not defined for negative numbers. Please enter a non-negative integer.")
    else:
        factorial = 1
        for i in range(1, num + 1):
            factorial *= i
        print(f"The factorial of {num} is {factorial}.")
except ValueError:
    print("Invalid input. Please enter a valid non-negative integer.")
```

Output:-



```
3.py
Enter a non-negative integer: 5
The factorial of 5 is 120.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & D:/annaconda/python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab
3.py
Enter a non-negative integer: 6
The factorial of 6 is 720.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
```

Justification:-

while I am giving the prompt, the first sentence I wrote , write a python code then automatically the sentence are visible, but again I write for then again it change. So , I observe in this which ever or whatever you are write or type It will change according to what you want to give the prompt will generated.

Task 2:

AI Code Optimization & Cleanup (Improving Efficiency)

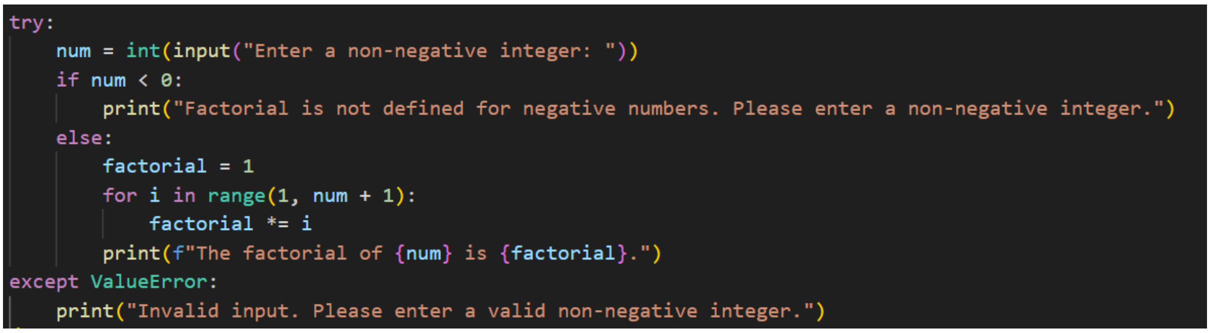
Prompt :-

#write a python code for factorial of a number without using function

#for factorial of a number without using function and tell if we type a negative number it should not calculate factorial and tell the user give a string input for these scenarios check the code below

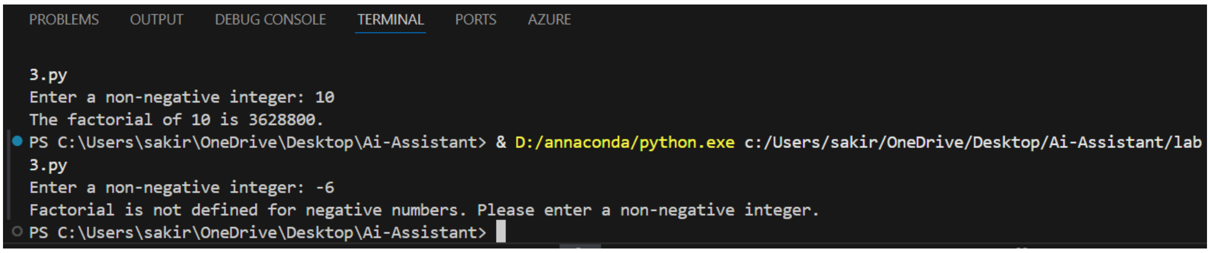
#if the user give a string input it should handle the error and tell the user to give a valid input.

Code :-



```python
try:
    num = int(input("Enter a non-negative integer: "))
    if num < 0:
        print("Factorial is not defined for negative numbers. Please enter a non-negative integer.")
    else:
        factorial = 1
        for i in range(1, num + 1):
            factorial *= i
        print(f"The factorial of {num} is {factorial}.")
except ValueError:
    print("Invalid input. Please enter a valid non-negative integer.")
```

Output :-



```
3.py
Enter a non-negative integer: 10
The factorial of 10 is 3628800.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & D:/annaconda/python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab
3.py
Enter a non-negative integer: -6
Factorial is not defined for negative numbers. Please enter a non-negative integer.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
```

Justification :-

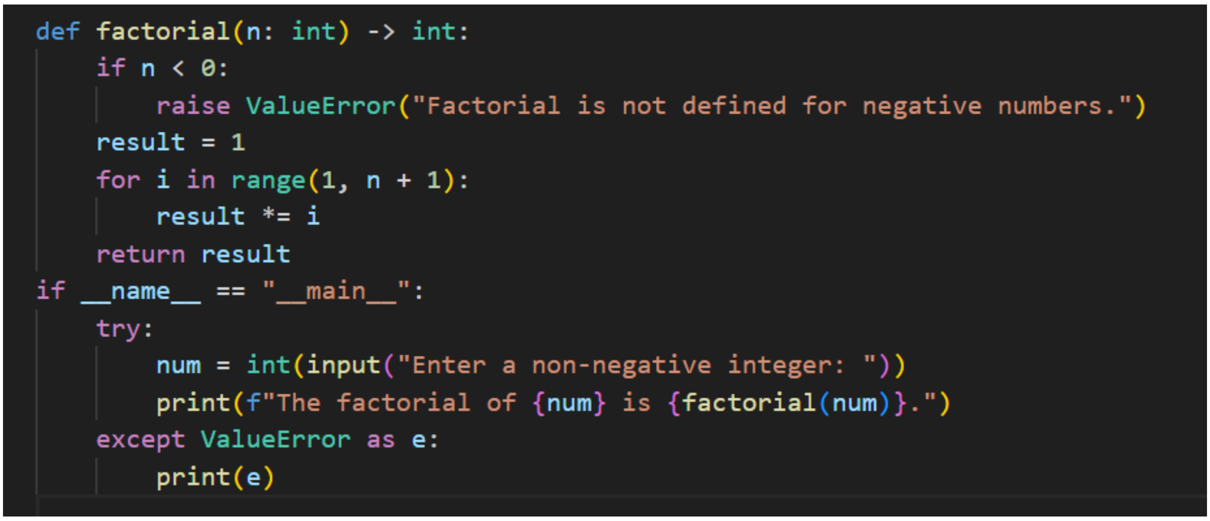
In this Task they gave about optimizing so I wrote if the given input is less then Zero(0) then print factorial is not defiend for negative numbers on that process I wrote the prompt to give the user input as a positive integer if accidently we press a letter or character then it shows invalid input please give correct input.

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

Prompt :-

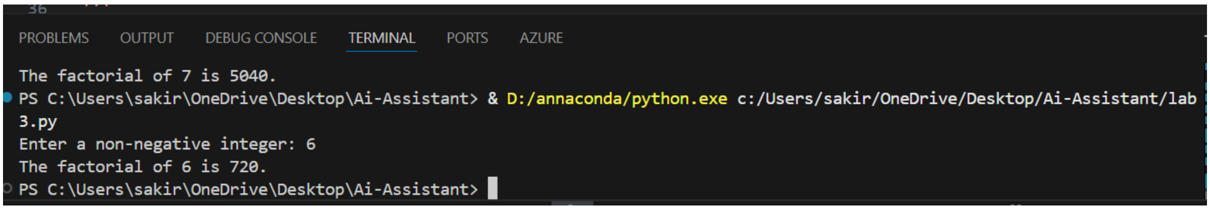
#write a python code for factorial of a number using function

Code :-



```python
def factorial(n: int) -> int:
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
if __name__ == "__main__":
    try:
        num = int(input("Enter a non-negative integer: "))
        print(f"The factorial of {num} is {factorial(num)}.")
    except ValueError as e:
        print(e)
```

Output :-



```
The factorial of 7 is 5040.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & D:/annaconda/python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab
3.py
Enter a non-negative integer: 6
The factorial of 6 is 720.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
```

Justification :-

In the above program I observe while using function the line code is reduced and it make optimized the code

---

Task 4:

Comparative Analysis – Procedural vs Modular AI Code (With vs Without Functions)

Difference:-

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.

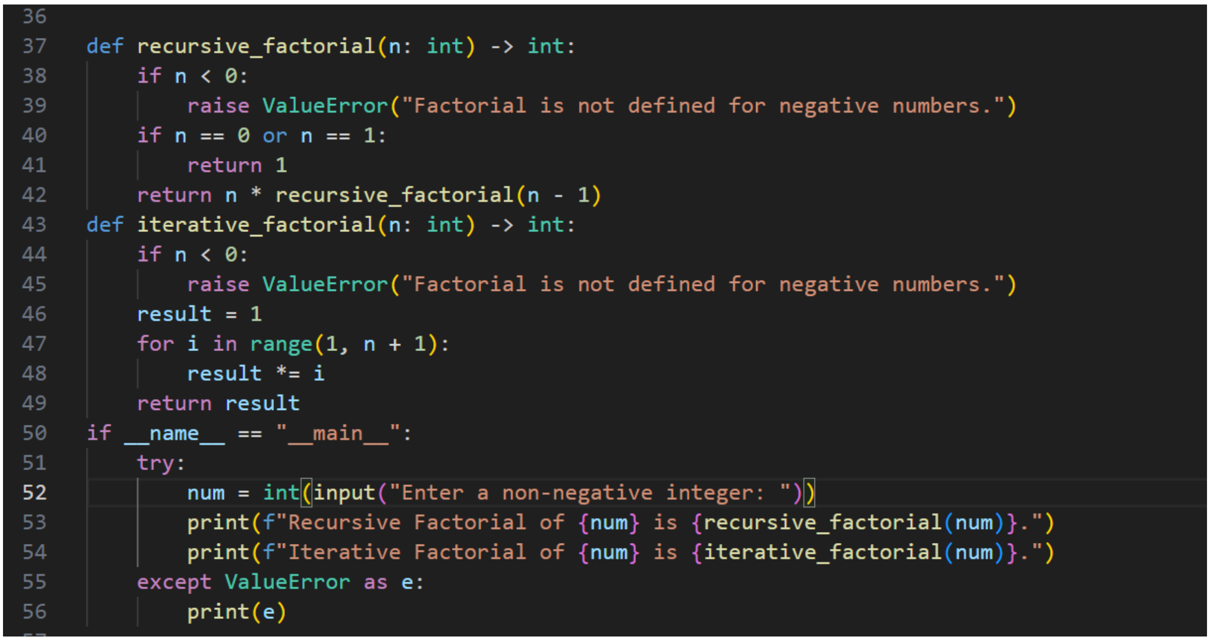
Although a factorial program is simple, using a **function-based (modular) approach** is more effective. It improves clarity, reusability, and maintainability while reducing long-term dependency on AI tools. Therefore, modular design is the preferred approach for both academic and real-world software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

Task-5:

AI-Generated Iterative vs Recursive Thinking

Prompt:- #write a program of python code to print factorial of a number using recursive and using iterative

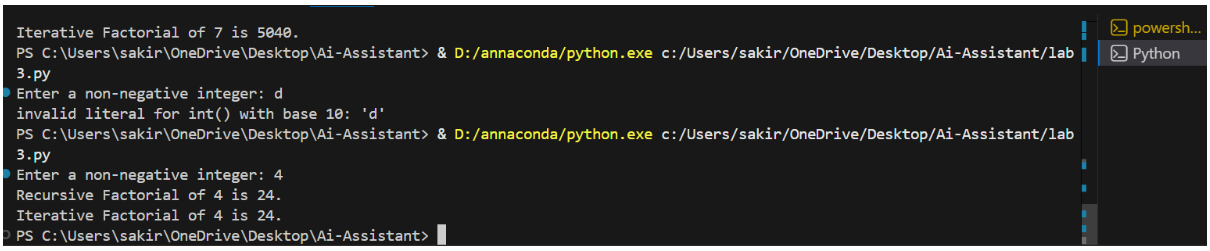
Code :-



```python

def recursive_factorial(n: int) -> int:
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    if n == 0 or n == 1:
        return 1
    return n * recursive_factorial(n - 1)
def iterative_factorial(n: int) -> int:
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
if __name__ == "__main__":
    try:
        num = int(input("Enter a non-negative integer: "))
        print(f"Recursive Factorial of {num} is {recursive_factorial(num)}.")
        print(f"Iterative Factorial of {num} is {iterative_factorial(num)}.")
    except ValueError as e:
        print(e)
```

Output:-



```
Iterative Factorial of 7 is 5040.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & D:/annaconda/python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab
3.py
Enter a non-negative integer: d
invalid literal for int() with base 10: 'd'
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & D:/annaconda/python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab
3.py
Enter a non-negative integer: 4
Recursive Factorial of 4 is 24.
Iterative Factorial of 4 is 24.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
```

Justification:-

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.