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

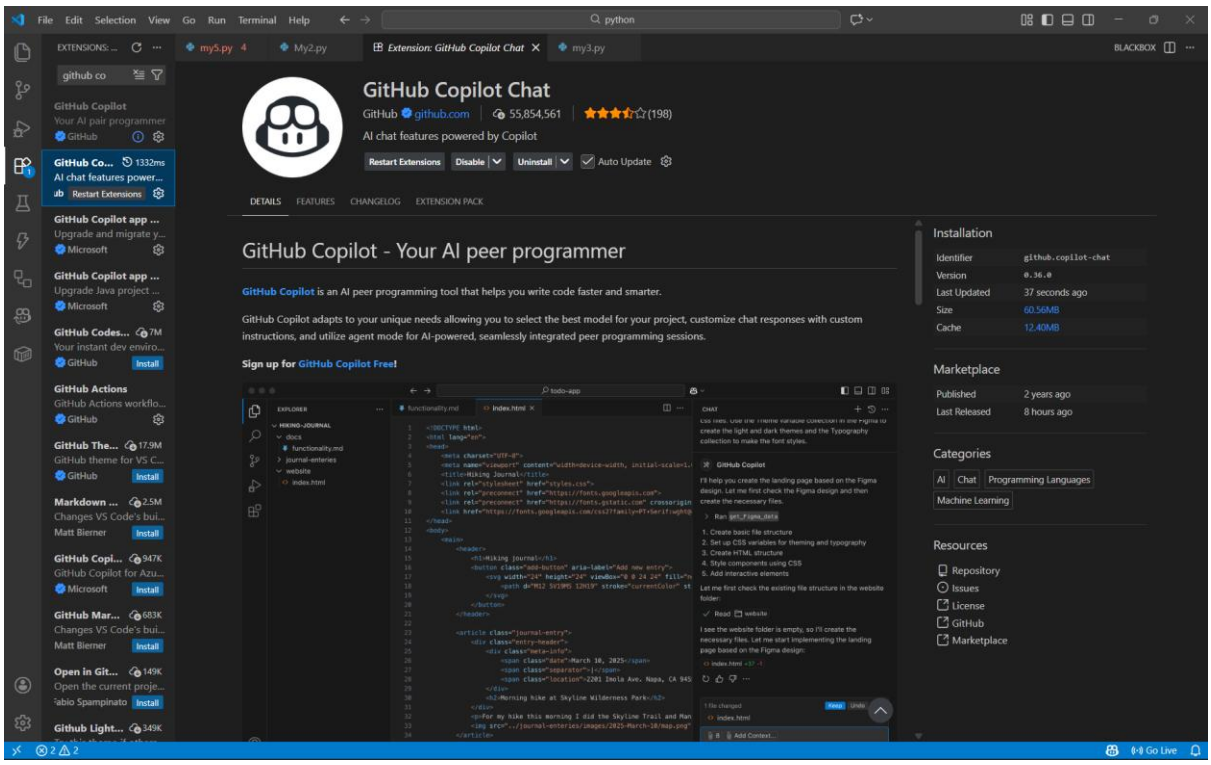
Name of Student : R.Deekshith

Enrollment No. : 2303A52104

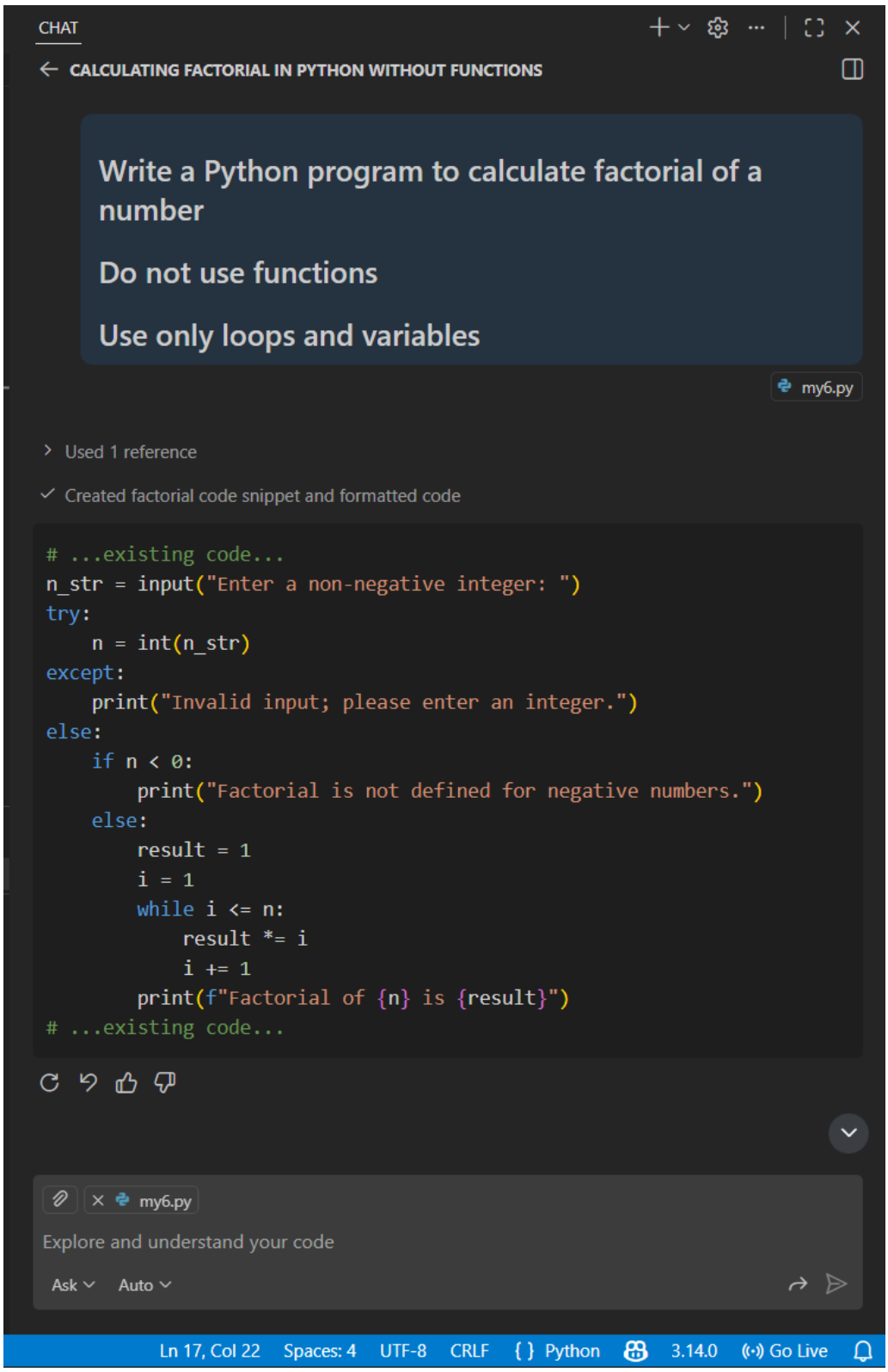
Batch No. : 33

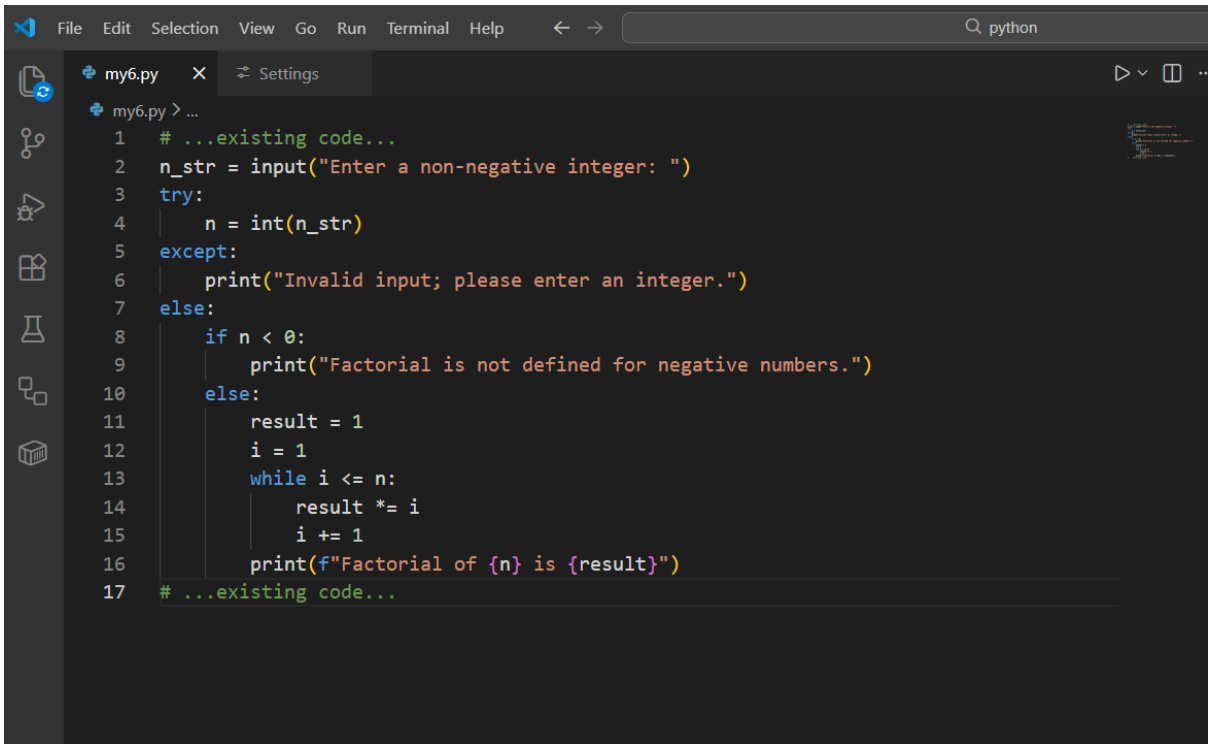
Date : 09/01/26

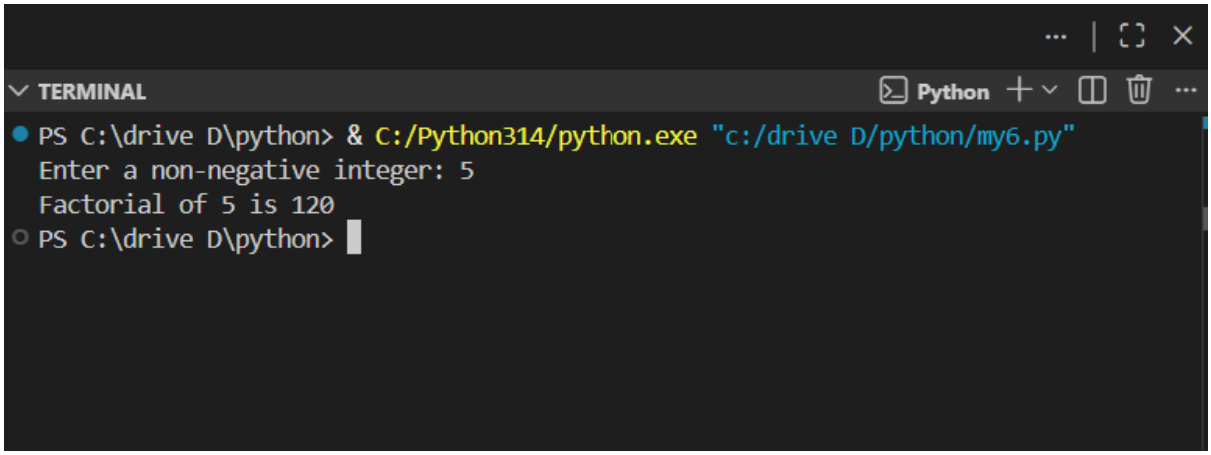
Git Hub


GitHub Copilot Chat
GitHub github.com 55,854,561 (198)
AI chat features powered by Copilot
Restart Extensions Disable Uninstall Auto Update
DETAILS FEATURES CHANGELOG EXTENSION PACK
GitHub Copilot - Your AI peer programmer
GitHub Copilot is an AI peer programming tool that helps you write code faster and smarter.
GitHub Copilot adapts to your unique needs allowing you to select the best model for your project, customize chat responses with custom instructions, and utilize agent mode for AI-powered, seamlessly integrated peer programming sessions.
Sign up for GitHub Copilot Free!
Installation
Identifier github.copilot-chat
Version 0.36.0
Last Updated 37 seconds ago
Size 60.56MB
Cache 12.40MB
Marketplace
Published 2 years ago
Last Released 8 hours ago
Categories
AI Chat Programming Languages Machine Learning
Resources
Repository
Issues
License
GitHub
Marketplace

TASK _01


CHAT
CALCULATING FACTORIAL IN PYTHON WITHOUT FUNCTIONS
Write a Python program to calculate factorial of a number
Do not use functions
Use only loops and variables
my6.py
Used 1 reference
Created factorial code snippet and formatted code
# ...existing code...
n_str = input("Enter a non-negative integer: ")
try:
n = int(n_str)
except:
print("Invalid input; please enter an integer.")
else:
if n < 0:
print("Factorial is not defined for negative numbers.")
else:
result = 1
i = 1
while i <= n:
result *= i
i += 1
print(f"Factorial of {n} is {result}")
# ...existing code...
my6.py
Explore and understand your code
Ask
Auto
Ln 17, Col 22
Spaces: 4
UTF-8
CRLF
{ } Python
3.14.0
Go Live





EXPLANATION:

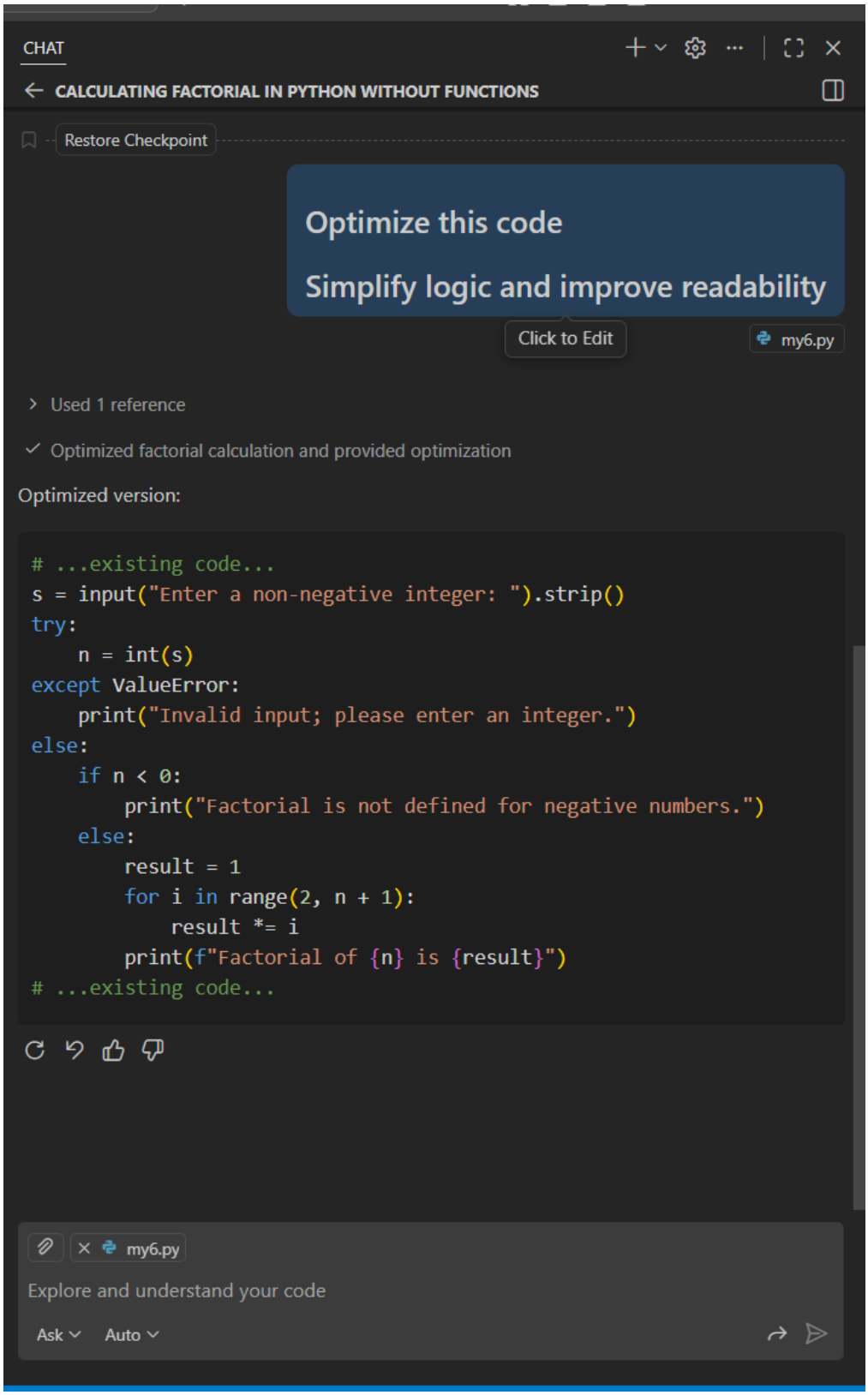
GitHub Copilot helped me quickly generate a working factorial program.

It followed basic Python syntax and logic correctly.

For beginners, it saves time and reduces syntax errors.

However, understanding the logic is still important.
Copilot gives suggestions but human review is needed

TASK_02:


CHAT
CALCULATING FACTORIAL IN PYTHON WITHOUT FUNCTIONS
Restore Checkpoint
Optimize this code
Simplify logic and improve readability
Click to Edit
my6.py
Used 1 reference
Optimized factorial calculation and provided optimization
Optimized version:
# ...existing code...
s = input("Enter a non-negative integer: ").strip()
try:
n = int(s)
except ValueError:
print("Invalid input; please enter an integer.")
else:
if n < 0:
print("Factorial is not defined for negative numbers.")
else:
result = 1
for i in range(2, n + 1):
result *= i
print(f"Factorial of {n} is {result}")
# ...existing code...
my6.py
Explore and understand your code
Ask
Auto

my6.py Settings

my6.py > ...

```python
s = input("Enter a non-negative integer: ").strip()
try:
    n = int(s)
except ValueError:
    print("Invalid input; please enter an integer.")
else:
    if n < 0:
        print("Factorial is not defined for negative numbers.")
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        print(f"Factorial of {n} is {result}")
```

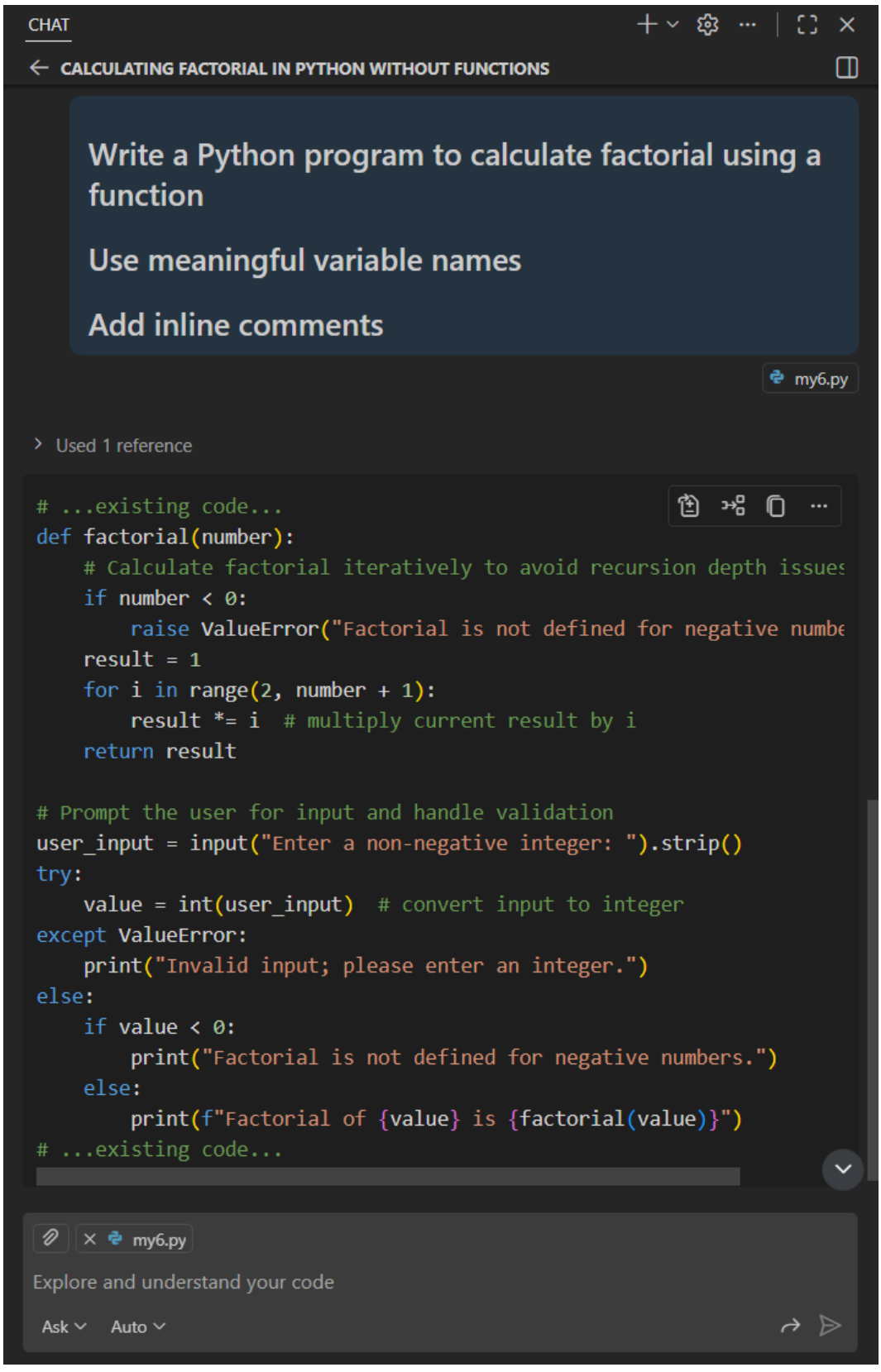
TERMINAL Python

```
PS C:\drive D\python> & C:/Python314/python.exe "c:/drive D/python/my6.py"
Enter a non-negative integer: 5
Factorial of 5 is 120
PS C:\drive D\python> & C:/Python314/python.exe "c:/drive D/python/my6.py"
Enter a non-negative integer: 7
Factorial of 7 is 5040
PS C:\drive D\python>
```

Explanation

- Better variable names
- Cleaner output
- More readable

TASK_03:


CHAT
CALCULATING FACTORIAL IN PYTHON WITHOUT FUNCTIONS
Write a Python program to calculate factorial using a function
Use meaningful variable names
Add inline comments
my6.py
Used 1 reference
# ...existing code...
def factorial(number):
# Calculate factorial iteratively to avoid recursion depth issues
if number < 0:
raise ValueError("Factorial is not defined for negative numbe
result = 1
for i in range(2, number + 1):
result *= i # multiply current result by i
return result
# Prompt the user for input and handle validation
user_input = input("Enter a non-negative integer: ").strip()
try:
value = int(user_input) # convert input to integer
except ValueError:
print("Invalid input; please enter an integer.")
else:
if value < 0:
print("Factorial is not defined for negative numbers.")
else:
print(f"Factorial of {value} is {factorial(value)}")
# ...existing code...
my6.py
Explore and understand your code
Ask
Auto

my6.py Settings

my6.py > ...

```python
# ...existing code...
def factorial(number):
    # Calculate factorial iteratively to avoid recursion depth issues
    if number < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    result = 1
    for i in range(2, number + 1):
        result *= i  # multiply current result by i
    return result

# Prompt the user for input and handle validation
user_input = input("Enter a non-negative integer: ").strip()
try:
    value = int(user_input)  # convert input to integer
except ValueError:
    print("Invalid input; please enter an integer.")
else:
    if value < 0:
        print("Factorial is not defined for negative numbers.")
    else:
        print(f"Factorial of {value} is {factorial(value)}")
# ...existing code...
```

EXPLANATION:

Using functions improves reusability.

The same function can be used in many programs.

Code becomes cleaner and easier to maintain.

TASK_04:

Comparative Analysis – Procedural vs Modular AI Code

Procedural (Without Functions) vs Modular (With Functions)

In Task 1, the factorial program was written using a procedural approach, where all the logic was implemented directly in the main execution flow without using any user-defined functions. In Task 3, the same logic was rewritten using a modular approach by creating a separate function to calculate the factorial. Both approaches produce the same output, but they differ significantly in terms of design quality and usability.

Logic Clarity:
The procedural version is simple and easy to understand for small programs. However, as the program grows, the logic becomes harder to follow because everything is written in one place. In contrast, the modular version separates the factorial logic into a function, making the code more organized and easier to read.

Reusability:
The procedural code cannot be reused easily because the logic is tied to a single script. The modular version allows the factorial function to be reused in multiple programs without rewriting the same code, which saves time and effort.

Debugging Ease:
Debugging procedural code is more difficult because errors can affect the entire program. In modular code, each function can be tested separately, making it easier to find and fix errors.

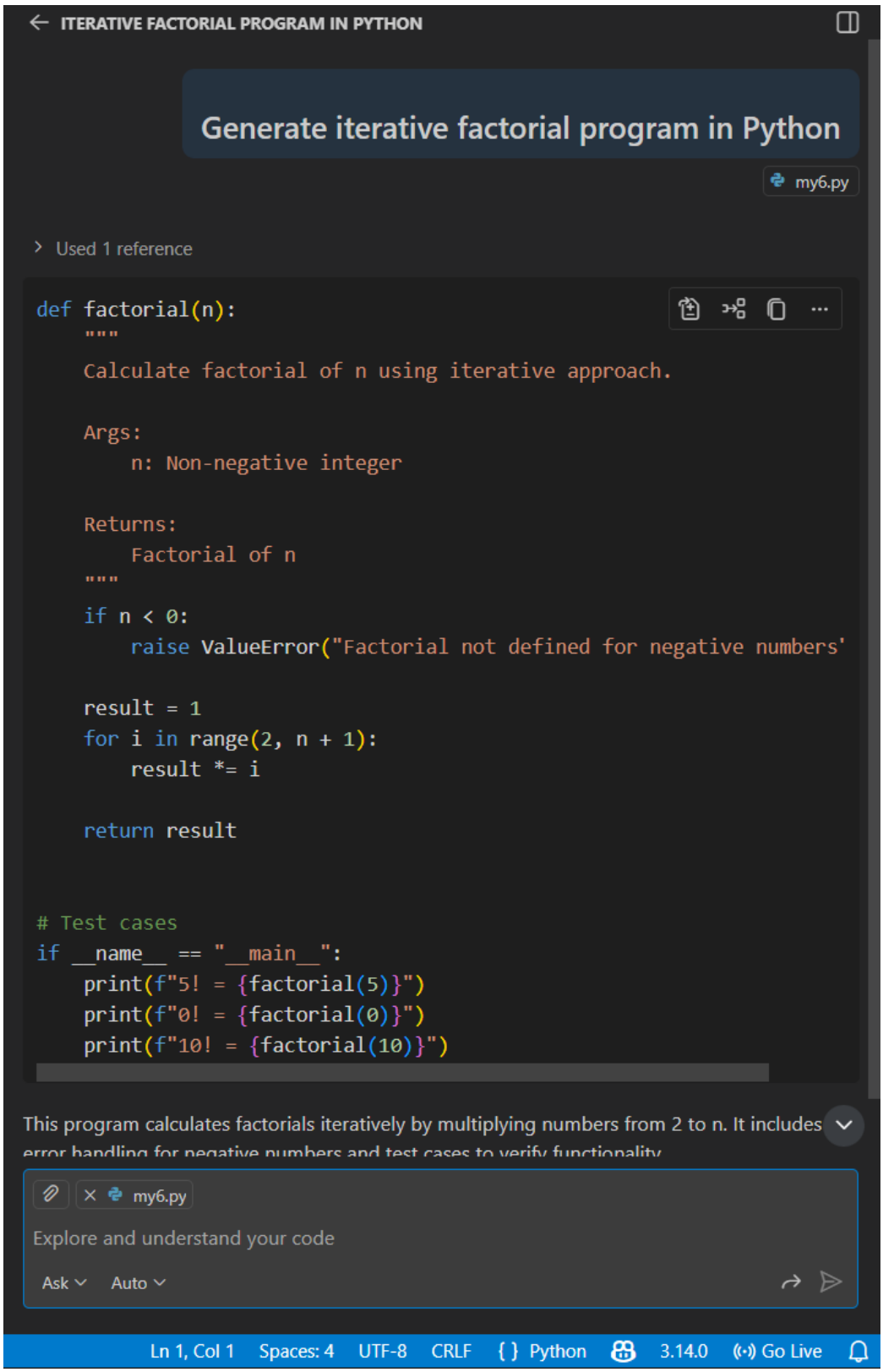
Suitability for Large Projects:
Procedural code is suitable only for small, simple programs. For large projects, modular code is preferred because it supports better structure, teamwork, and maintenance.

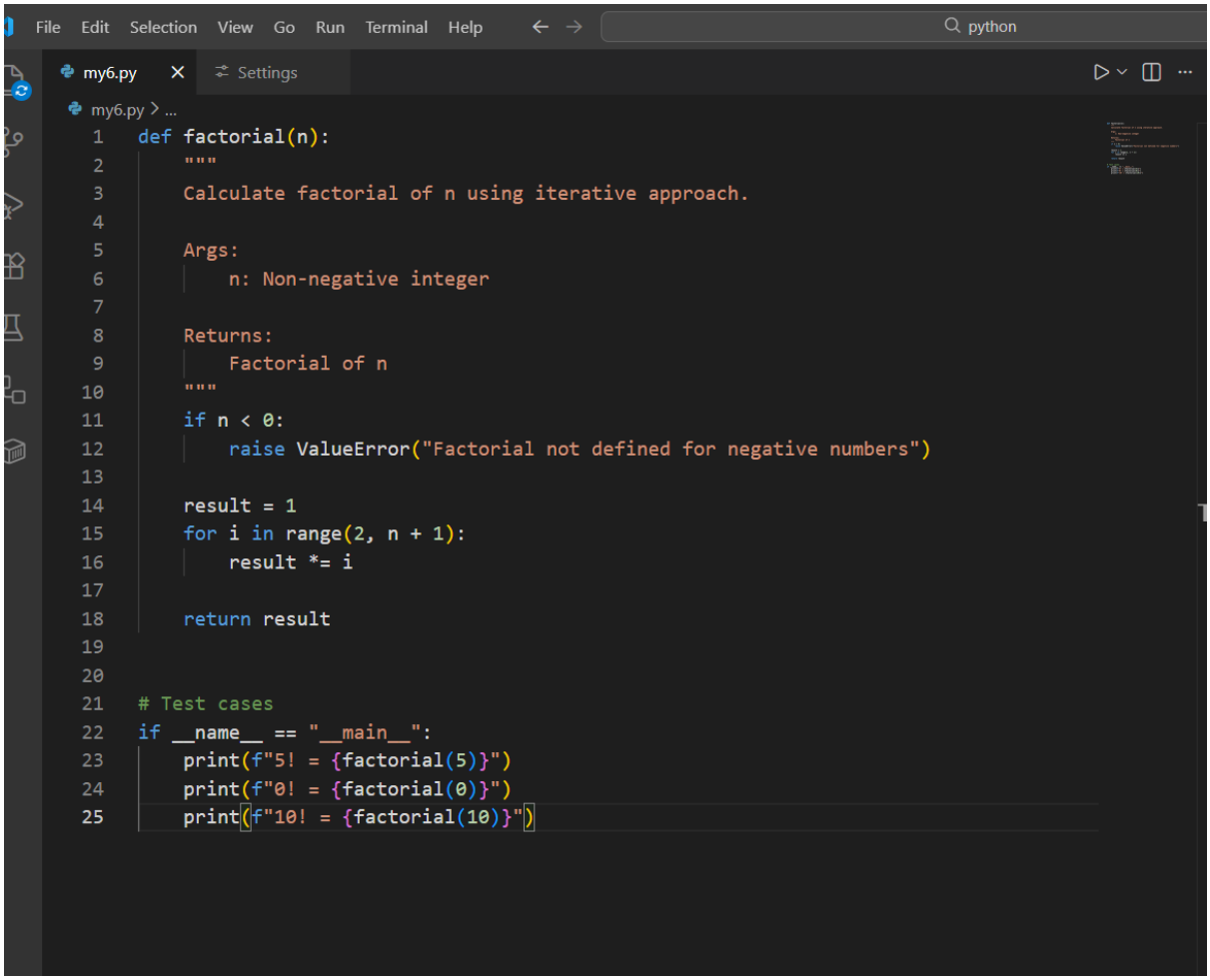
AI Dependency Risk:
When using AI tools like GitHub Copilot, procedural code may be generated quickly but often lacks proper structure. Modular code encourages better design practices, even when AI is used. This reduces the risk of poor-quality code.

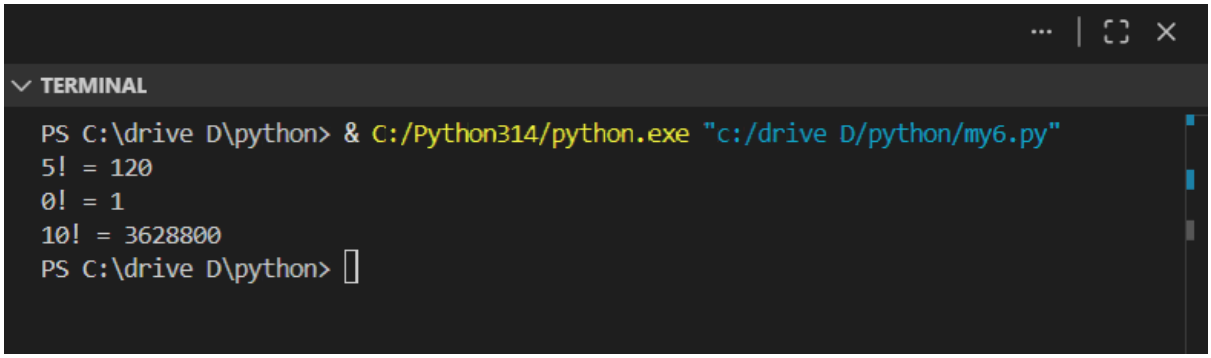
Conclusion:

While procedural programming is useful for quick tasks and learning basics, modular programming is more efficient, reusable, and suitable for real-world software development. Using functions improves clarity, maintainability, and scalability, making modular code the better choice for professional projects.

TASK_05:


ITERATIVE FACTORIAL PROGRAM IN PYTHON
Generate iterative factorial program in Python
my6.py
Used 1 reference
def factorial(n):
"""
Calculate factorial of n using iterative approach.
Args:
n: Non-negative integer
Returns:
Factorial of n
"""
if n < 0:
raise ValueError("Factorial not defined for negative numbers'
result = 1
for i in range(2, n + 1):
result *= i
return result
# Test cases
if __name__ == "__main__":
print(f"5! = {factorial(5)}")
print(f"0! = {factorial(0)}")
print(f"10! = {factorial(10)}")
This program calculates factorials iteratively by multiplying numbers from 2 to n. It includes
my6.py
Explore and understand your code
Ask
Auto
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
{ } Python
3.14.0
Go Live





Explanation

- Iterative uses loop
- Recursive calls itself
- Recursion uses more memory
- Iterative is faster