# **Lab 10: Code Review and Quality: Using AI to improve code quality and readability**

Assignment number: 10.2

Enrollment number:2503A51L36

Name: Suhana Rehan

**Lab Objectives:**

* To understand the importance of code readability, maintainability, and quality.
* To explore how AI-assisted coding tools can review code and suggest improvements.
* To practice identifying code smells, redundant code, and poor naming conventions.
* To apply AI tools for refactoring and improving readability.
* To critically evaluate AI feedback and integrate it into real projects

**Lab Outcomes (LOs):**

After completing this lab, students will be able to:

* Use AI-assisted tools (e.g., GitHub Copilot, Cursor AI) to review Python code.
* Identify and correct syntax issues, code smells, and inefficient logic.
* Improve readability by applying consistent formatting, naming, and comments.
* Refactor code with AI suggestions while ensuring functionality is preserved.
* Apply best practices for writing clean, maintainable, and professional code.

**Task 1: AI-Assisted Code Review (Basic Errors)**

* Write python program as shown below.
* Use an AI assistant to review and suggest corrections

**Prompt:** Review and suggest me correct code

def calcFact(n):

result = 1

x = 0

for i in range(1, n):

result = result \* i

return result

def main():

num = 5

FACT = calcFact(num)

print("the factorial of", num, "is", FACT)

t = 10

if FACT > 10:

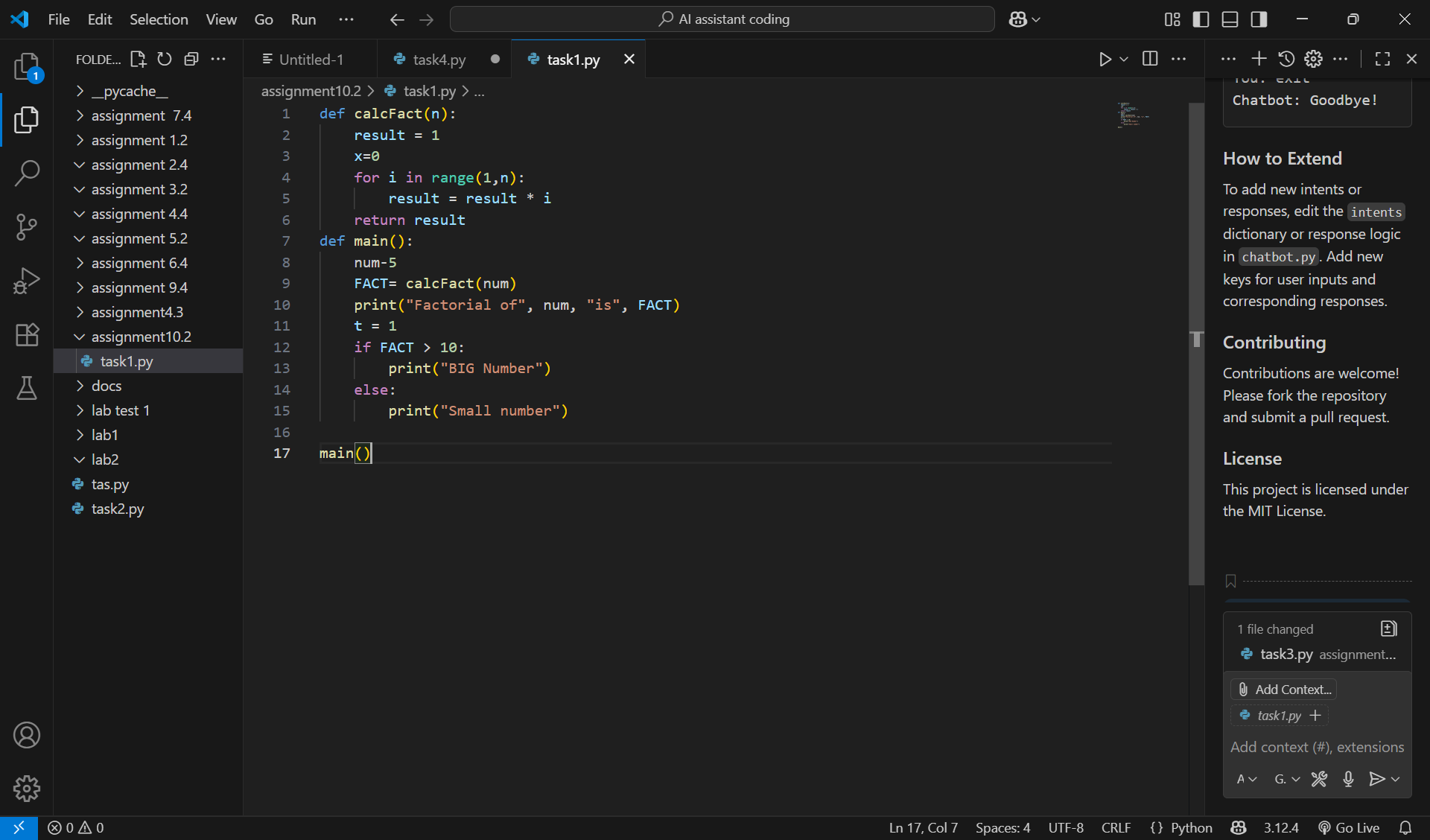
print("BIG Number")

else:

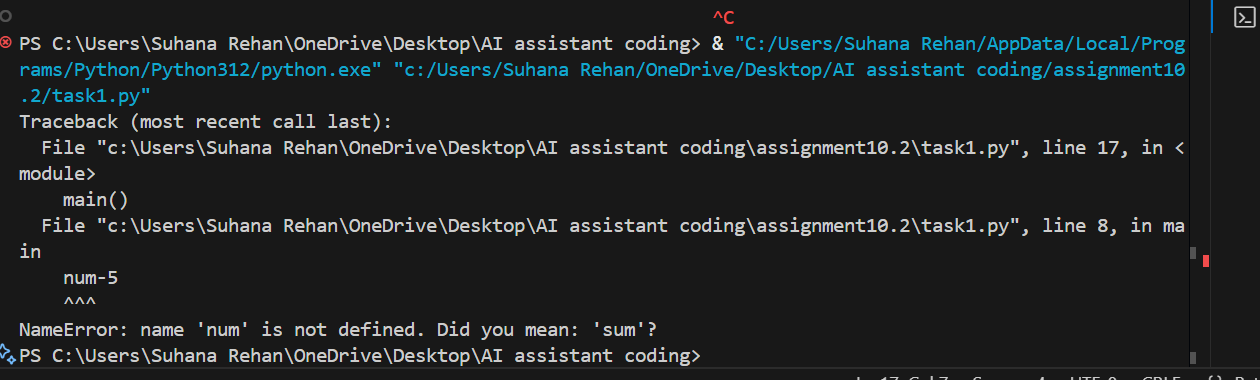
print("small number")

main()

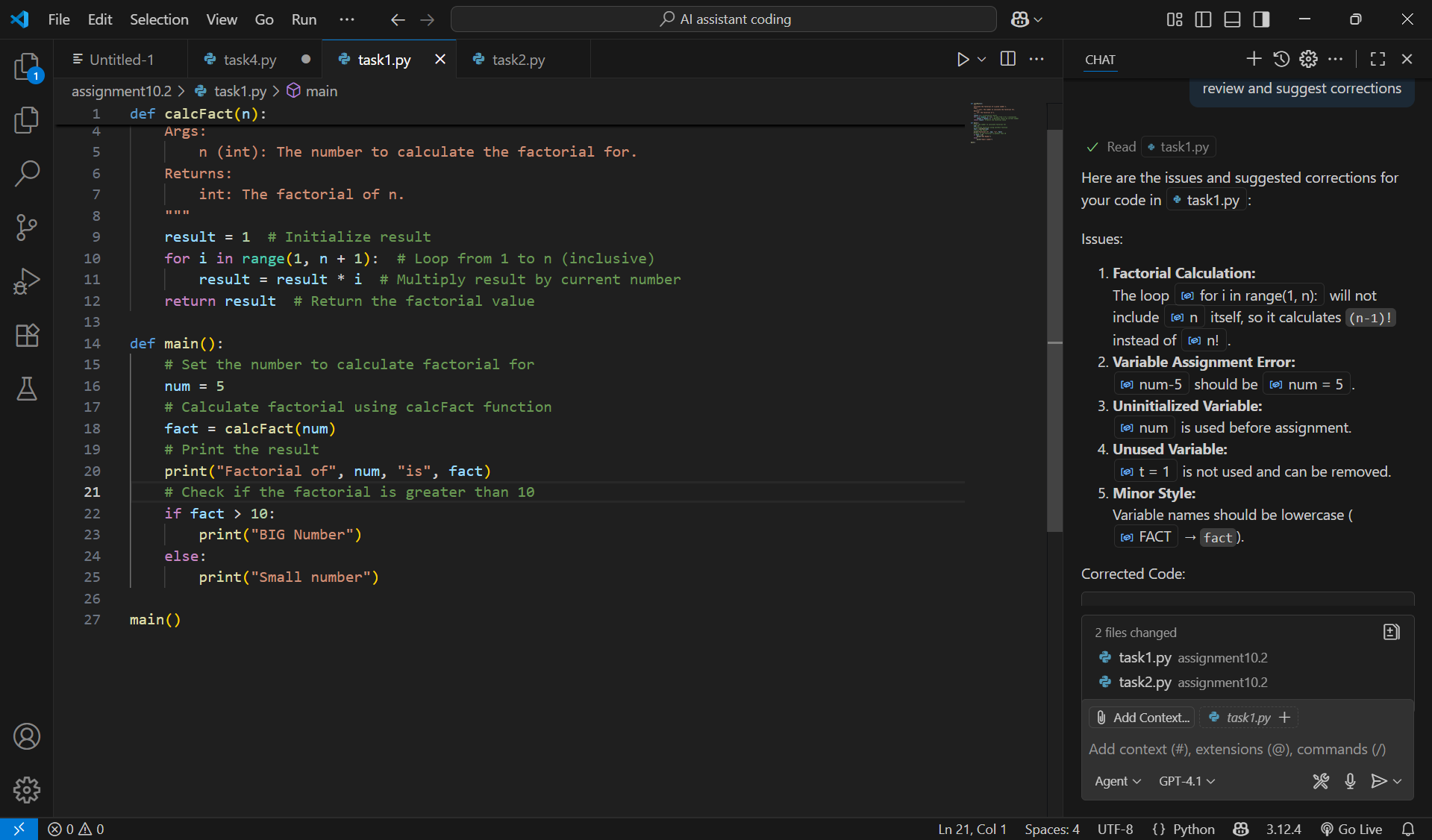
**Code without AI:**

****

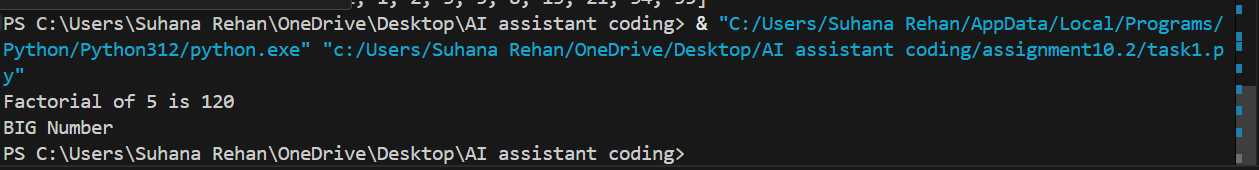
**OUPUT without AI:**

****

**Code After AI reviewed it:**

****

**OUPUT After AI reviewed it:**

****

**Observation:**

Here in this task ai reviewed the suggested the correct version

Of the code and it also added the comment line for better understanding

**Task 2: Automatic Inline Comments**

* Write the Python code for Fibonacci as shown below and execute.
* Ask AI to improve variable names, add comments, and apply PEP8 formatting (cleaned up).
* Students evaluate which suggestions improve readability most. one.

**Prompt:**

def f1(xx):

a = 0

b = 1

c = 2

Zz = [a, b]

while c <= xx:

d = a + b

Zz.append(d)

a = b

b = d

c = c + 1

return Zz

def m():

NN = 10

ans = f1(NN)

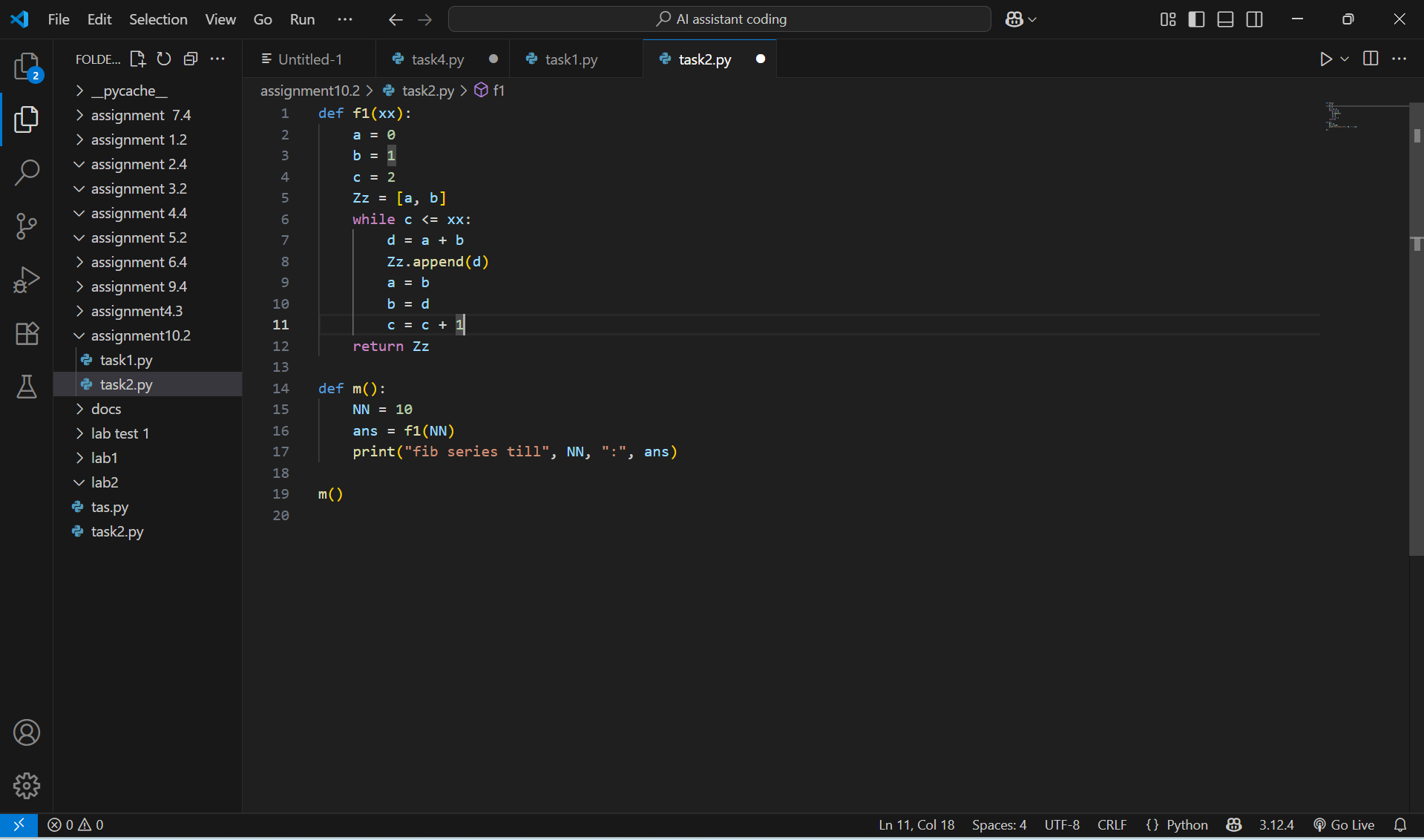
print("fib series till", NN, ":", ans)

m()

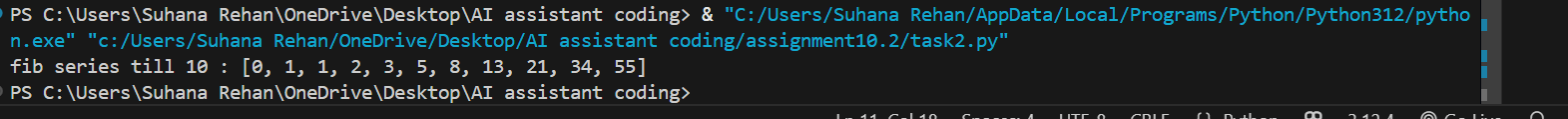
**Prompt for AI:**

Improve variable names, add comments, and apply PEP8 formatting

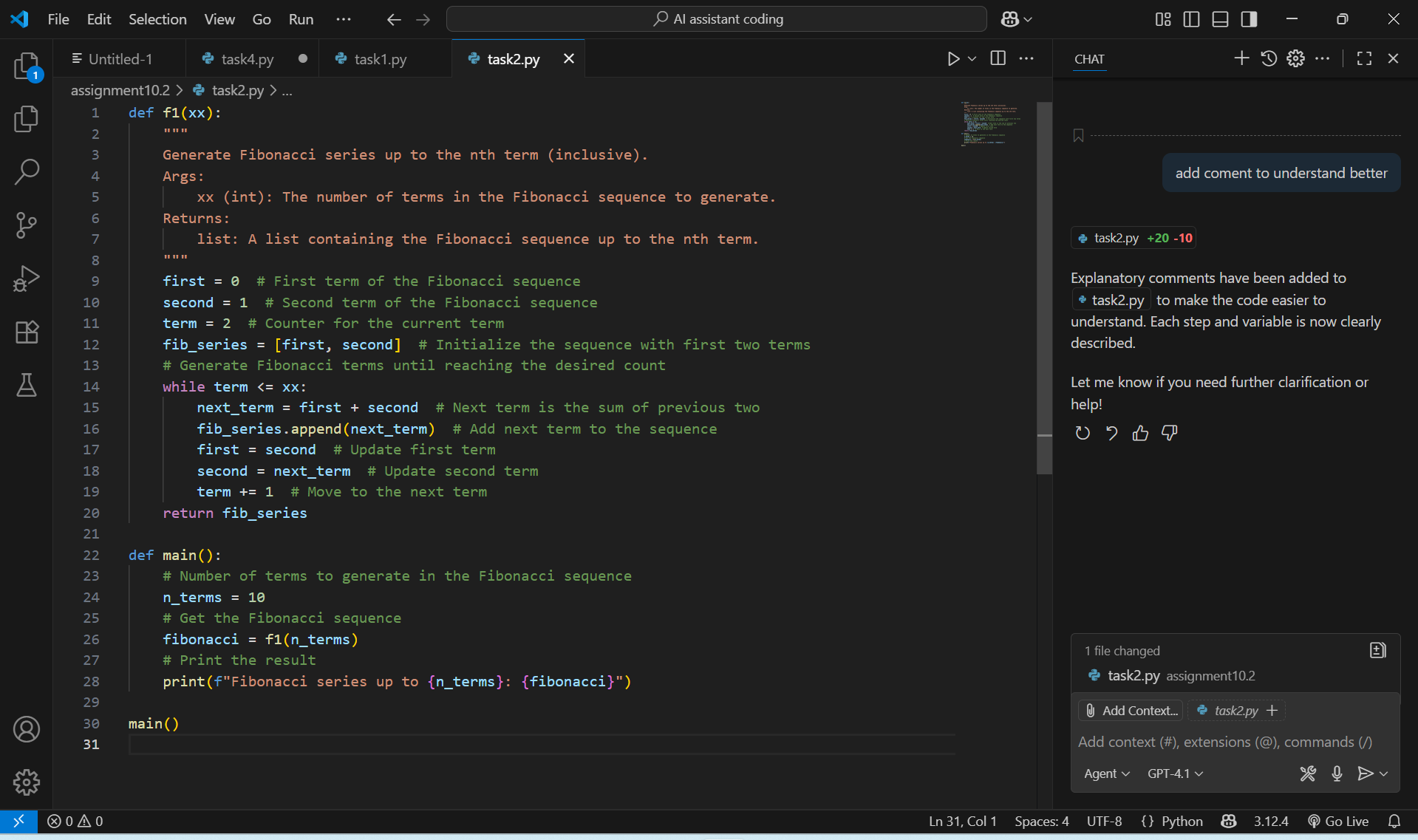
**Code without AI:**



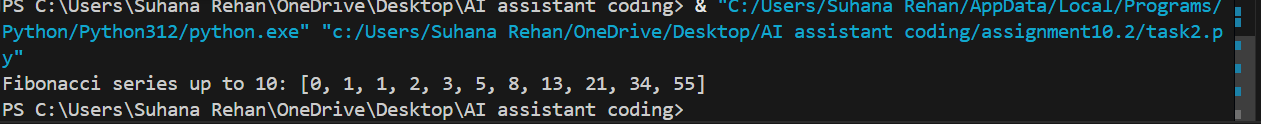
**Output Without AI:**



**Code with AI:**



**Output With AI:**

****

**Observation:**

**AI** made the code the easy to read and understand and correct the code. AI used PEP8 Format.AI added the comments and docstring to make the code more understandable and readable.

**Task 3:**

• Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).

• Incorporate manual docstring in code with NumPy Style

• Use AI assistance to generate a module-level docstring + individual function docstrings.

• Compare the AI-generated docstring with your manually written one.

**Prompt:**

"""

A simple calculator with basic operations: add, subtract, multiply, and divide.

"""

def add(a, b):

"""

Add two numbers and return the result.

"""

return a + b

def subtract(a, b):

"""

Subtract b from a and return the result.

"""

return a - b

def multiply(a, b):

"""

Multiply two numbers and return the result.

"""

return a \* b

def divide(a, b):

"""

Divide a by b and return the result.

Raises an error if b is zero.

"""

if b == 0:

raise ValueError("Cannot divide by zero")

return a / b

if \_\_name\_\_ == "\_\_main\_\_":

print("Add: ", add(10, 5))

print("Subtract: ", subtract(10, 5))

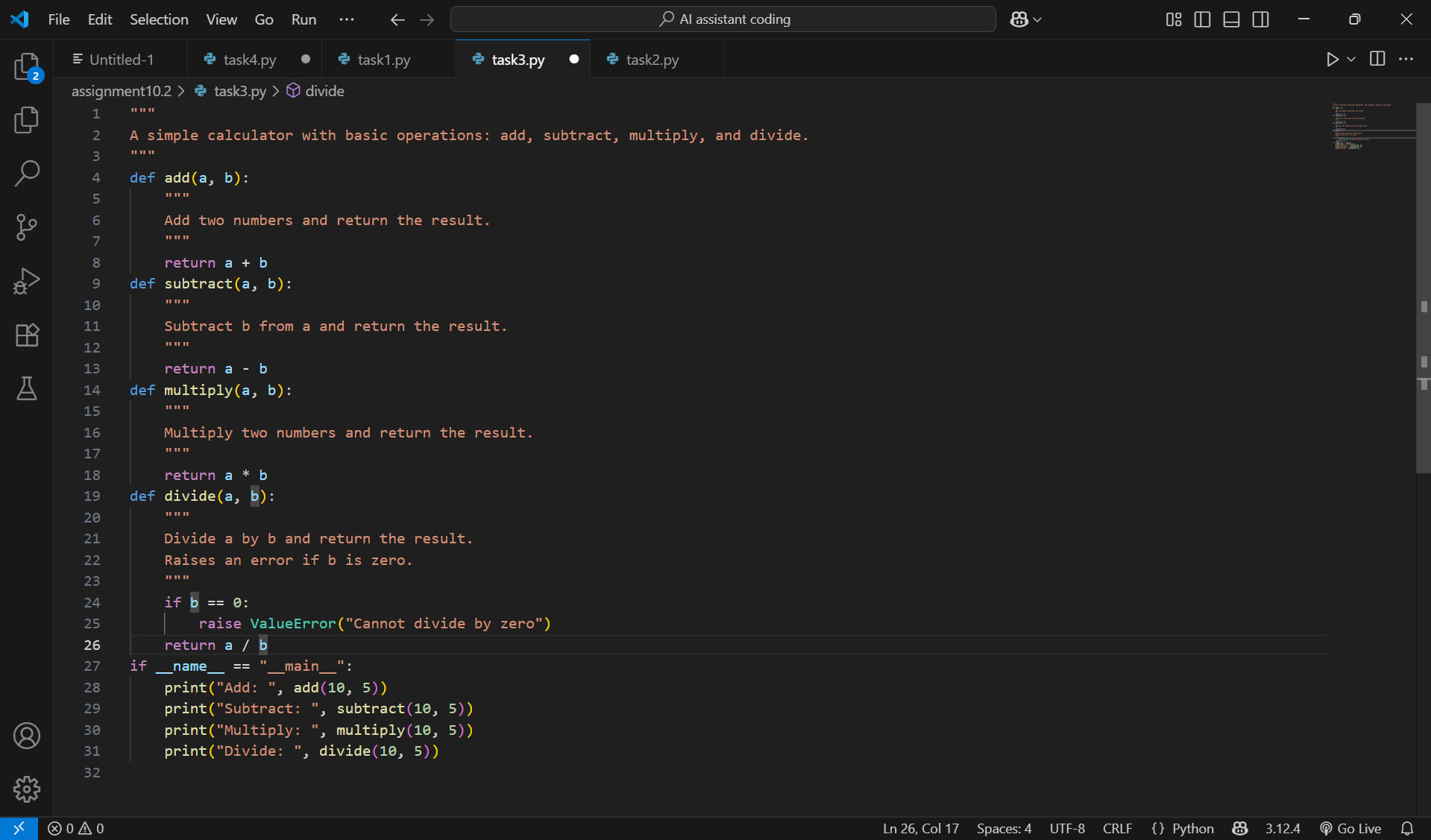
print("Multiply: ", multiply(10, 5))

print("Divide: ", divide(10, 5))

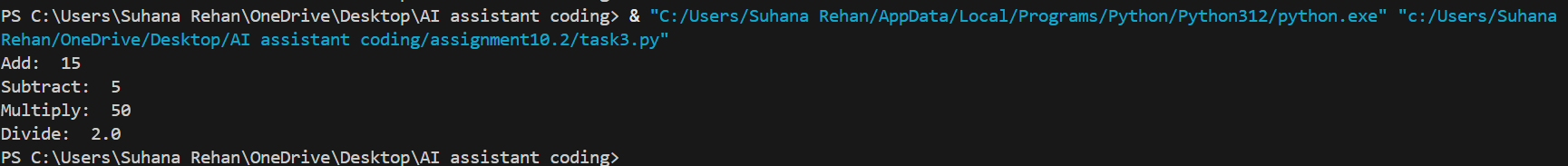
**Prompt for AI:**

generate a module-level docstring + individual function docstrings with an different example and make the output more polished

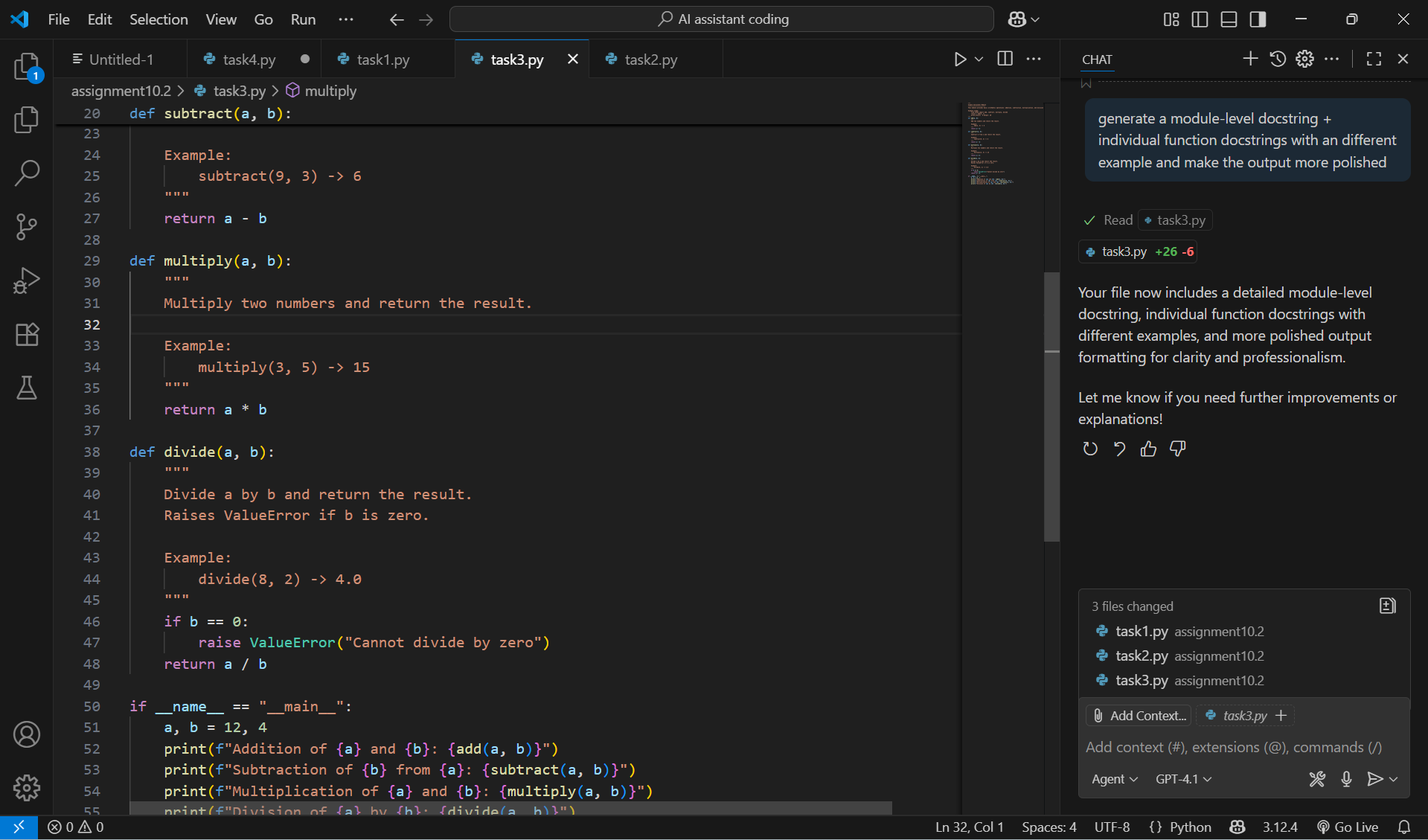
**Code without AI:**

****

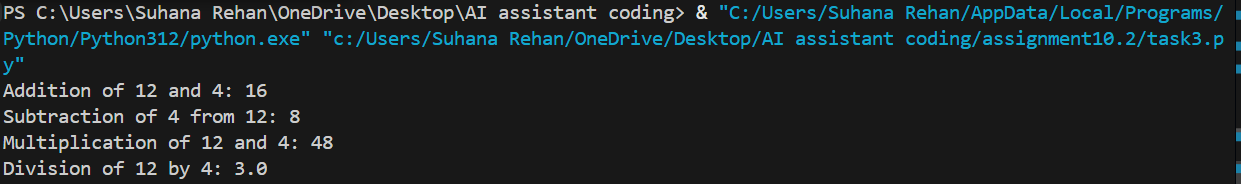
**Output Without AI:**

****

**Code with AI:**

****

**Output With AI:**

****

**Observation:**

* Manual docstrings are better when teaching or when clarity is needed for future maintainers.
* AI-generated docstrings are quicker to write and useful when documentation isn’t the primary focus but completeness is required.
* It's ideal to combine both approaches—start with AI-generated templates, then refine them manually.