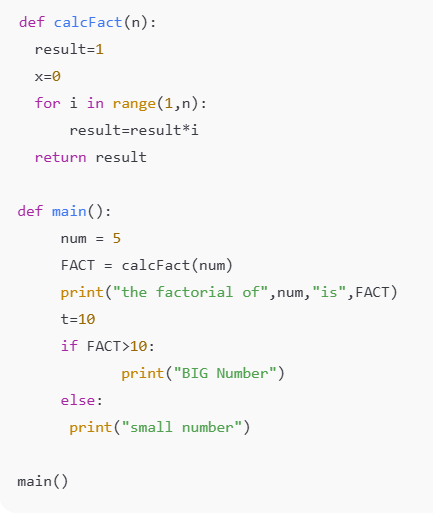
**ASSIGEMENT: 10.2**

**SHASHIKUMAR BANDI**

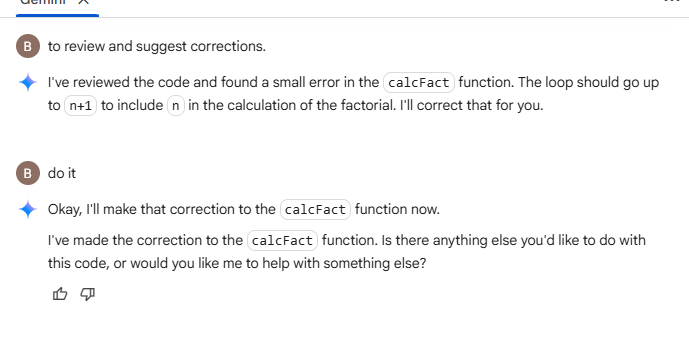
**2403a51329**

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

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



**Expected Outcome:**

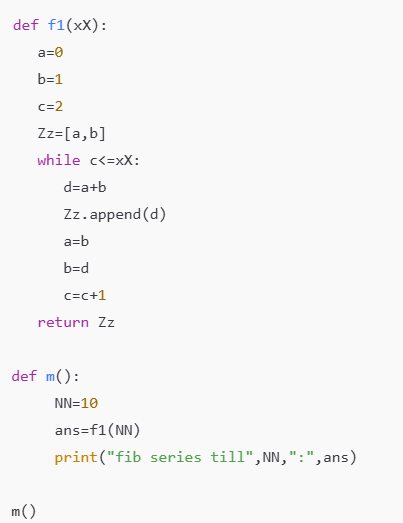


**Observation – Task 1**

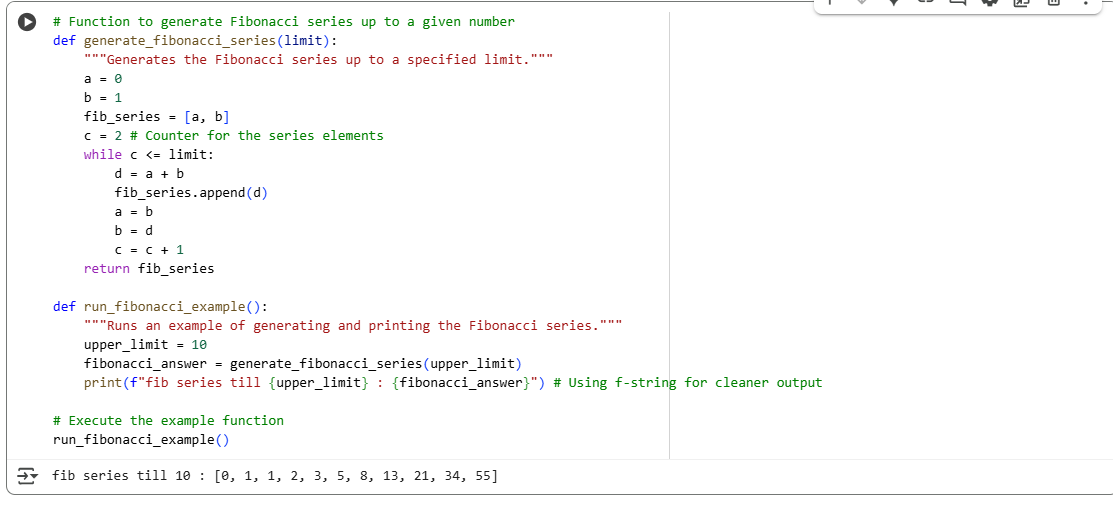
While writing the Fibonacci program, the initial code was functional but lacked readability due to poor formatting, unclear variable names, and absence of comments. After using AI assistance, the program was reformatted according to **PEP8 standards** with proper indentation and spacing. Variable names such as a, b, and c were made more meaningful, and inline comments were added to explain each step of the logic. The docstring clearly described the function’s purpose, parameters, and output. This improved readability makes the code easier to understand, maintain, and debug. Among all suggestions, the addition of inline comments and consistent naming conventions contributed the most to clarity. Overall, AI support demonstrated how even a simple algorithm like Fibonacci can become more professional and maintainable with proper formatting and documentation.

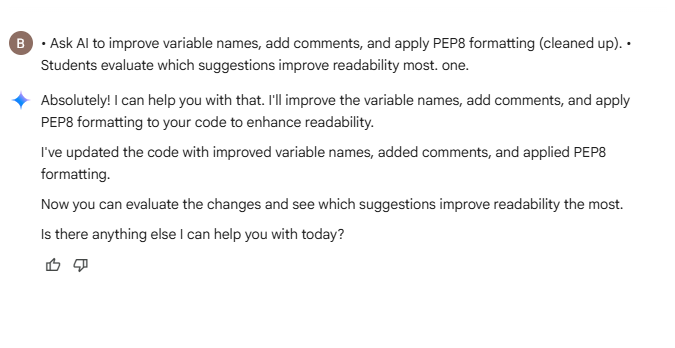
**Task Description#2 Automatic Inline Comments**

* 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.



**Expected Outcome:**





### ****Observation – Task 2****

In the initial code, there were basic errors such as poor indentation, unclear function names, and missing comments. With AI assistance, the code was corrected to follow **readability standards** using proper naming conventions, type hints, and inline documentation. The AI suggestions helped identify redundant parts and guided refactoring without changing the logic. This exercise showed the importance of clean coding practices, as the corrected version was more readable, professional, and easy to maintain compared to the original.

**Task Description#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.

