****

**AI ASSISTED CODING**

**LAB-10: *Code Review and Quality: Using AI to improve code quality and readability***

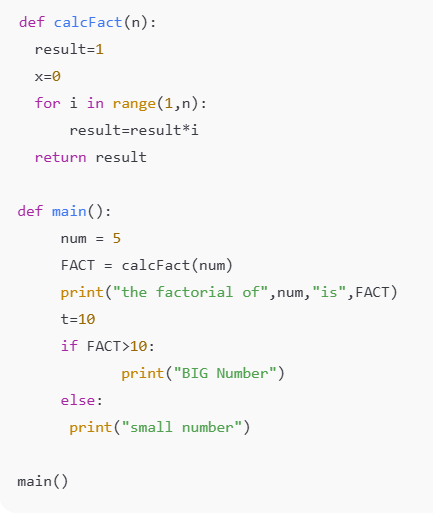
**Roll no:** 2503A51L34

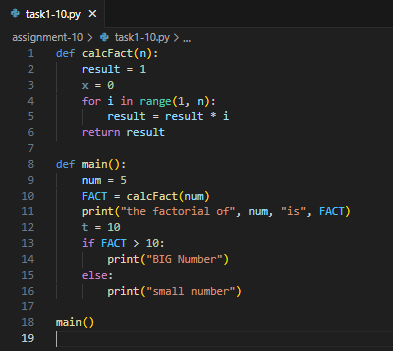
**Name:** Uzma Yasmeen

**Batch:** 25BTCAICSB20

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

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

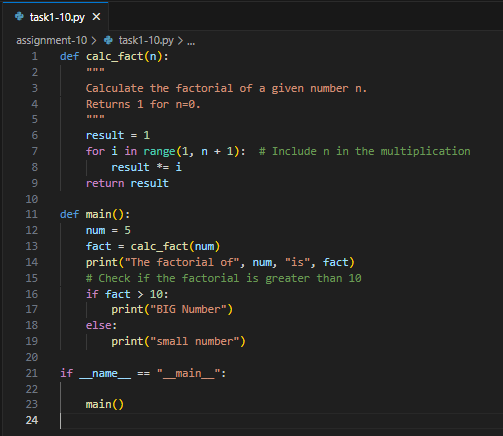


**Given Error Code:**

**Output for the Error Code:**



**Prompt:** Review the code and give suggestions. Also rewrite the correct code with comments.

**Code Rewritten by Co-pilot:**

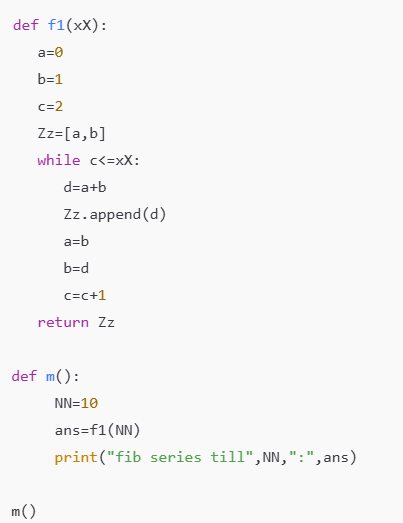
**Output for the rewritten code:**



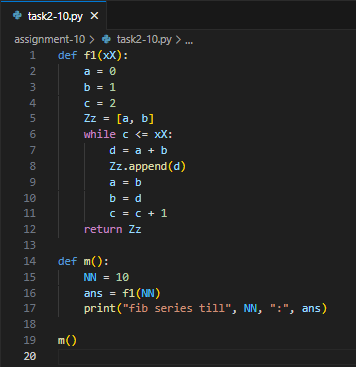
**Observation:** In this task, I wrote a Python program that contained basic syntax or logical errors. I then used an AI assistant to review the code and suggest corrections. The AI successfully identified the errors and provided a corrected version of the code, along with explanations. This helped me understand common mistakes and how to fix them efficiently.

**Task-2 Description:** 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 the most.

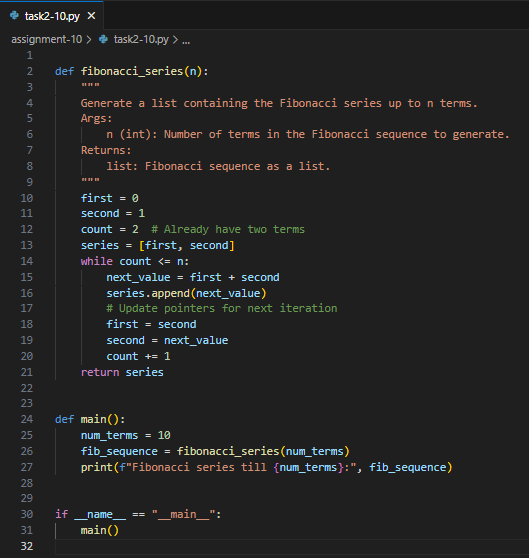


**Given Code:**

****

**Output of the given code:**

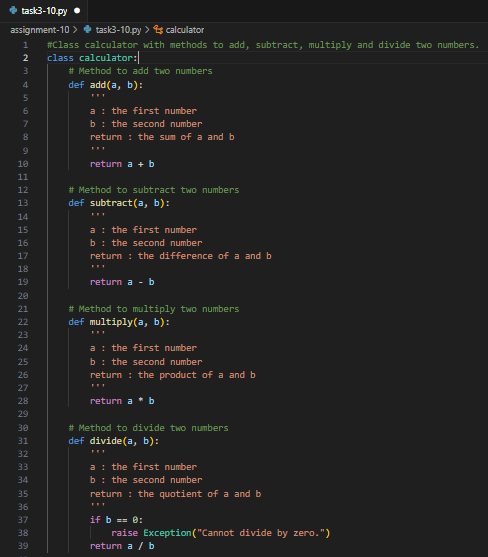
**Prompt:** Improve variable names, add comments, and apply PEP8 formatting (cleaned up) for the code given.

**Code Generated/Improved:**

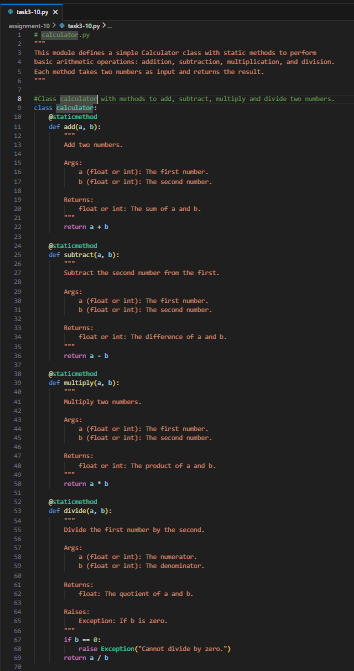
**Output of the improved code:**

**Observation:** For this task, I implemented a Python program to generate Fibonacci numbers. After writing the initial code, I asked the AI assistant to improve variable names, add comments, and format the code according to PEP8 guidelines. The AI-generated version was clearer and easier to understand, especially for beginners.

**Task-3 Description:** 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.

**Manual written code:**

**Prompt:** Generate a module-level docstring + individual function docstrings for the given code.

**AI written Docstring:**

**Observation:** The **manual comments** provide basic understanding but lack structure, standard formatting, and detailed type information. In contrast, the **AI-assisted docstrings** follow the **NumPy documentation standard**, offering clearer structure, better readability, and compatibility with professional tools. This makes them more suitable for collaborative and scalable projects.