NAME : DARIPELLI SRIYA

ENROLLMENT NO : 2403A51350

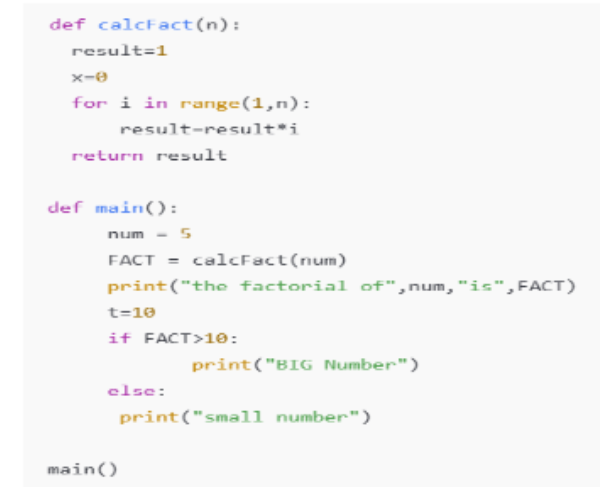
BATCH NO: 01

SUBJECT: AI ASSISTANT CODING

ASSIGNMENT: 10.2

QUESTION:

Task Description#1 AI-Assisted Code Review (Basic Errors)  
• Write python program as shown below.  
• Use an AI assistant to review and suggest corrections.

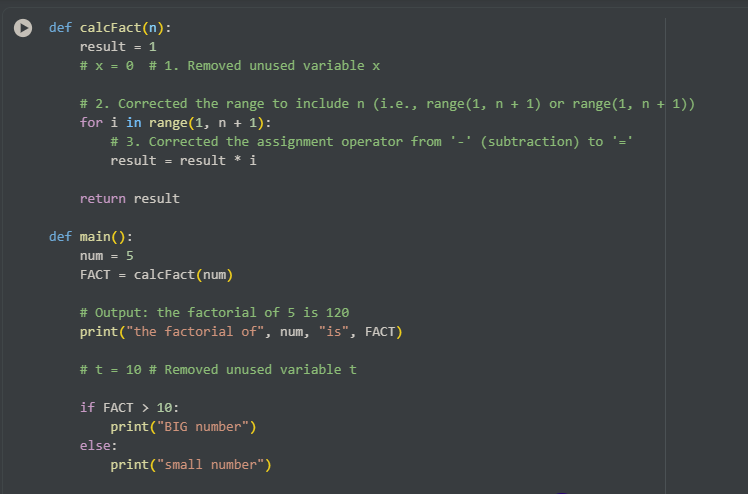


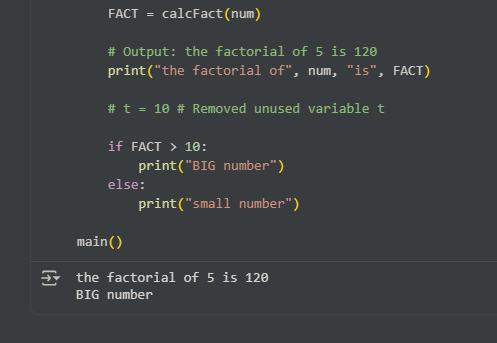
Expected Outcome#1: Students need to submit corrected code with comments.

PROMPT:

(PICTURE OS THIS) CORRECT THE CODE

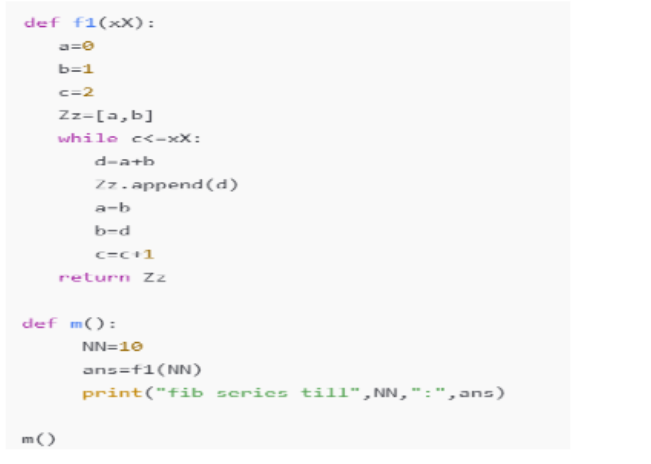
OUTPUT SCREENSHOTS:





EXPLAINATION: IT IS OBSERVED THAT THE MAIN MISTAKE THE LOOP IS ITERATED UPTO N-1 TIMES BUT IT NEEDS TO ITERATE N TIMES

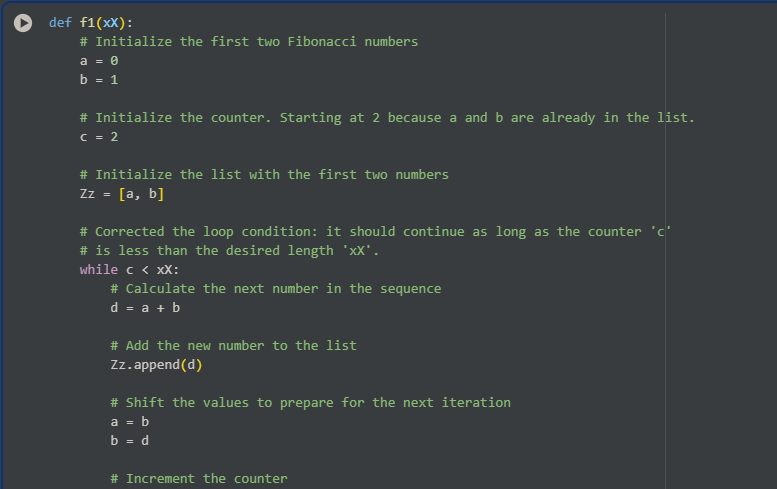
Task Description#2 Automatic Inline Comments  
• Write the Python code for Fibonacci as shown below and execute.  
• Ask AI to improve variable names, add comments, and apply PEP8 formatting  
(cleaned up).  
• Students evaluate which suggestions improve readability most. one.

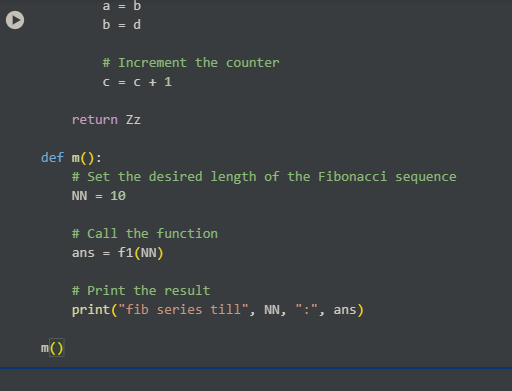
  
Expected Output#2: Clean format python code with much readability.

PROMPT:

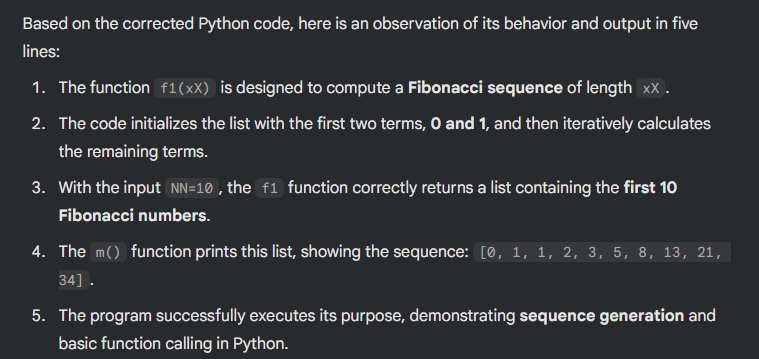
(PICTURE OS THIS) CORRECT THE CODE

SCREENSHOTS:





EXPLAINATION:



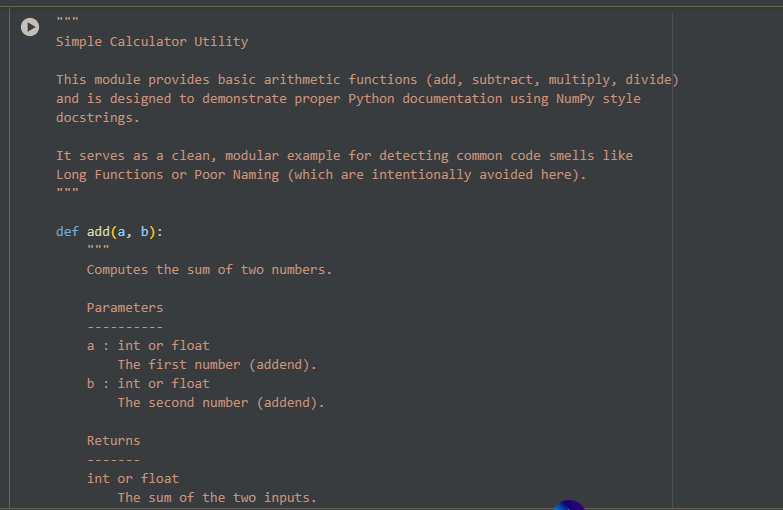
Task Description#3  
• Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply,  
divide).  
• Incorporate manual docstring in code with NumPy Style  
• Use AI assistance to generate a module-level docstring + individual function  
docstrings.  
• Compare the AI-generated docstring with your manually written one.  
Common Examples of Code Smells  
• Long Function – A single function tries to do too many things.  
• Duplicate Code – Copy-pasted logic in multiple places.  
• Poor Naming – Variables or functions with confusing names (x1, foo, data123).  
• Unused Variables – Declaring variables but never using them.  
• Magic Numbers – Using unexplained constants (3.14159 instead of PI).  
• Deep Nesting – Too many if/else levels, making code hard to read.  
• Large Class – A single class handling too many responsibilities.  
Why Detecting Code Smells is Important  
• Makes code easier to read and maintain.  
• Reduces chance of bugs in future updates.  
• Helps in refactoring (improving structure without changing behavior).  
• Encourages clean coding practices  
Dead Code – Code that is never executed.

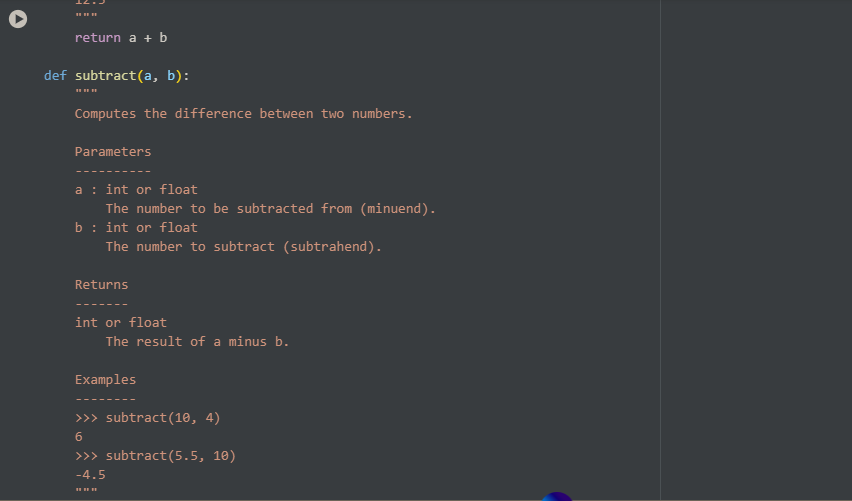
PROMPT:

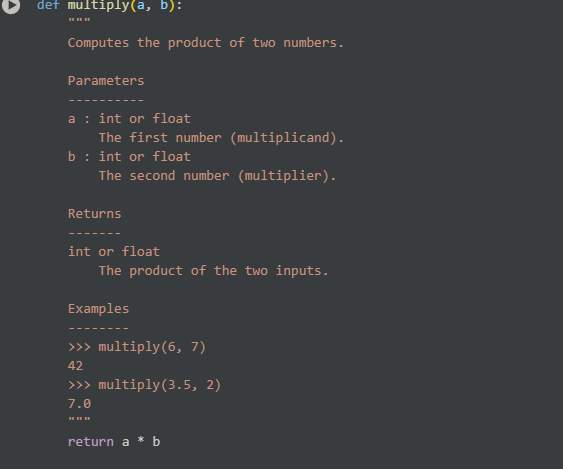
Confirm that the module-level docstring and all four function docstrings (`add`, `subtract`, `multiply`, `divide`) are correctly formatted using the Google Docstring Style (Args, Returns, Raises where applicable). 2.Analyze all four functions for the following code smells: 'Long Function' and 'Deep Nesting'. State whether the code is clean or requires refactoring.

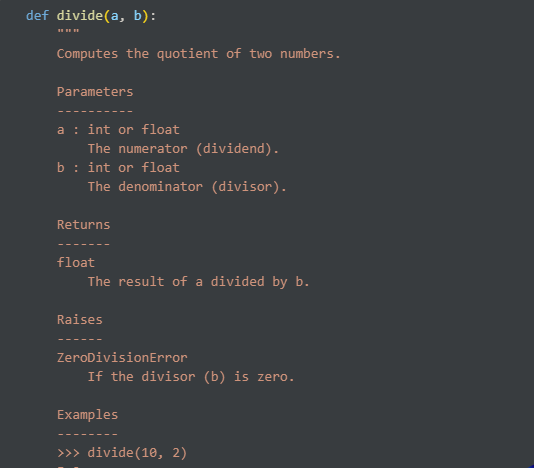
Expected Output#3: Students learn structured documentation for multi-function scripts

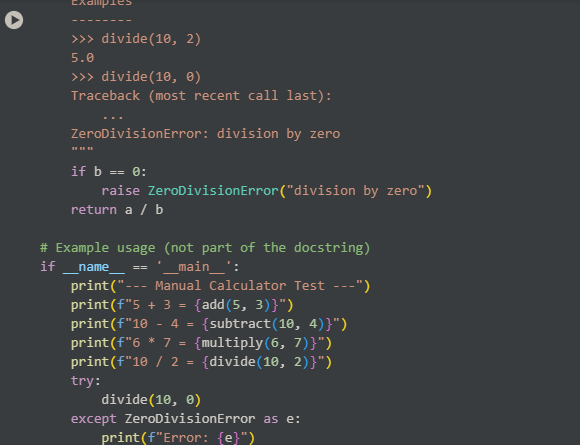
SCREENSHOTS:

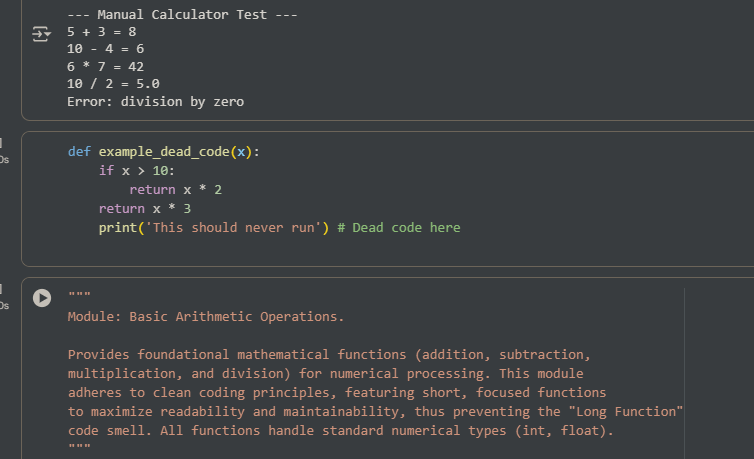


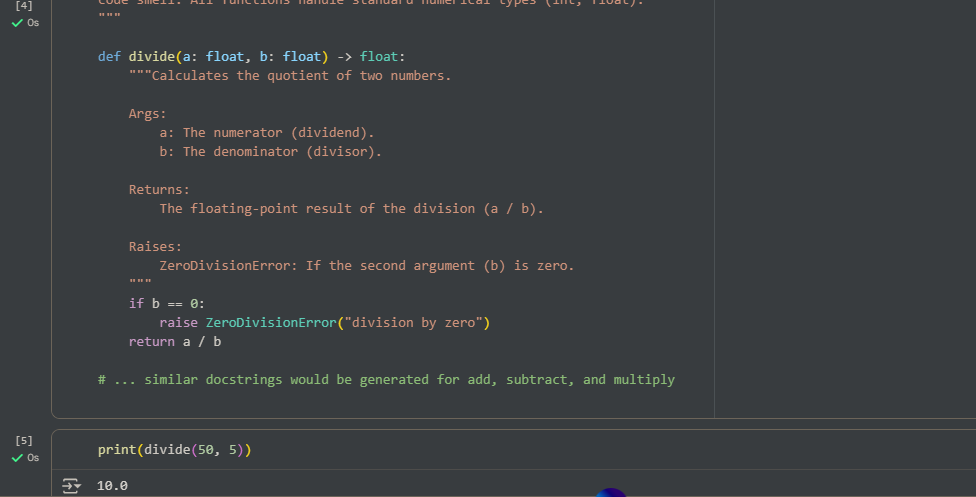


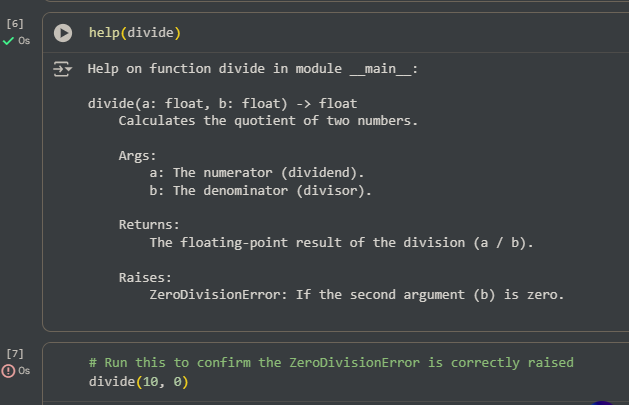


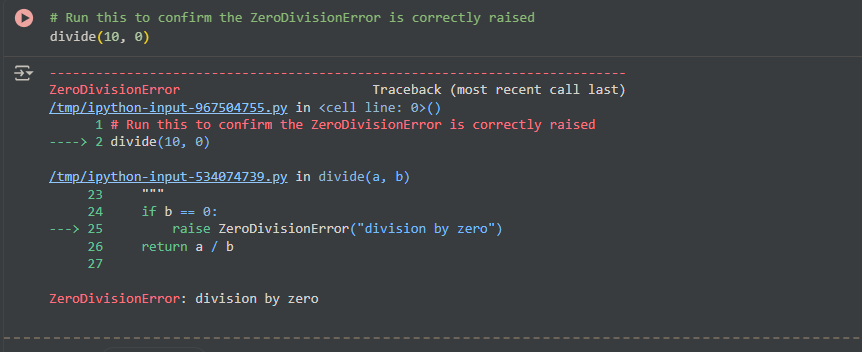












OBSERVATION:

