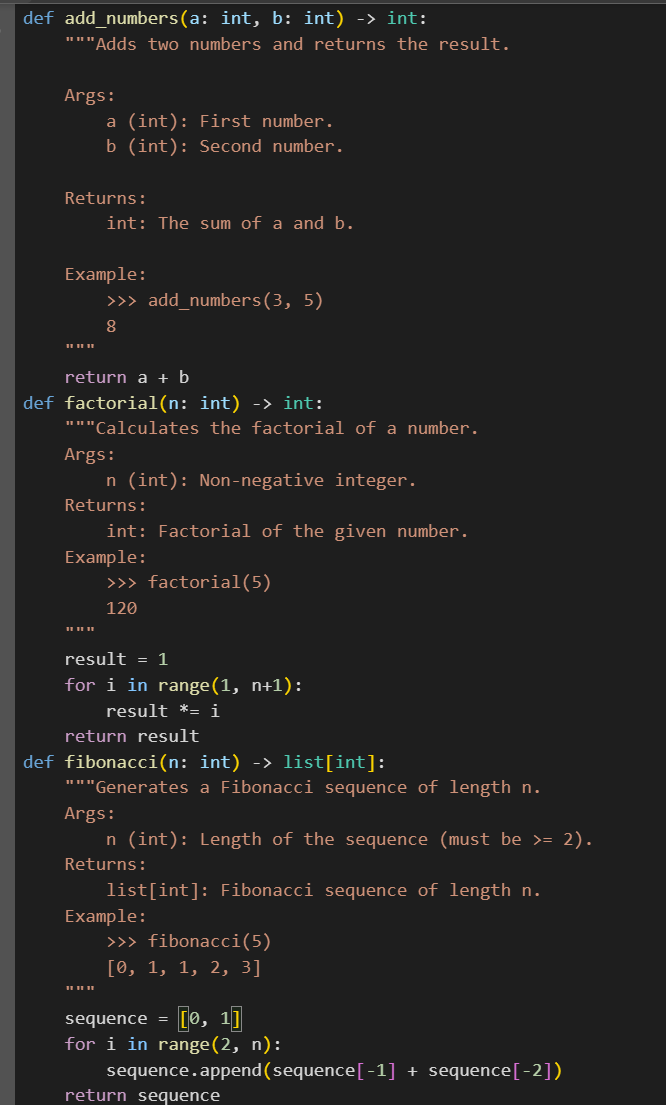
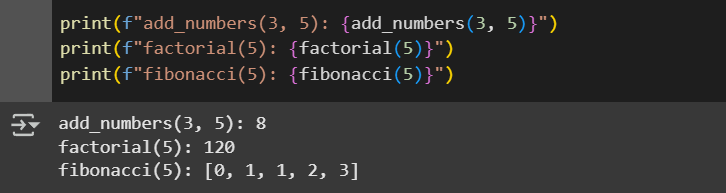
**Lab Assignment 9.1**

**Task Description #1 (Documentation – Google-Style Docstrings for  
Python Functions)  
• Task: Use AI to add Google-style docstrings to all functions in a  
given Python script.  
• Instructions:  
o Prompt AI to generate docstrings without providing any  
input-output examples.  
o Ensure each docstring includes:  
▪ Function description  
▪ Parameters with type hints  
▪ Return values with type hints  
▪ Example usage  
o Review the generated docstrings for accuracy and  
formatting.  
• Expected Output #1:  
o A Python script with all functions documented using  
correctly formatted Google-style docstrings**

**Code and Output:**

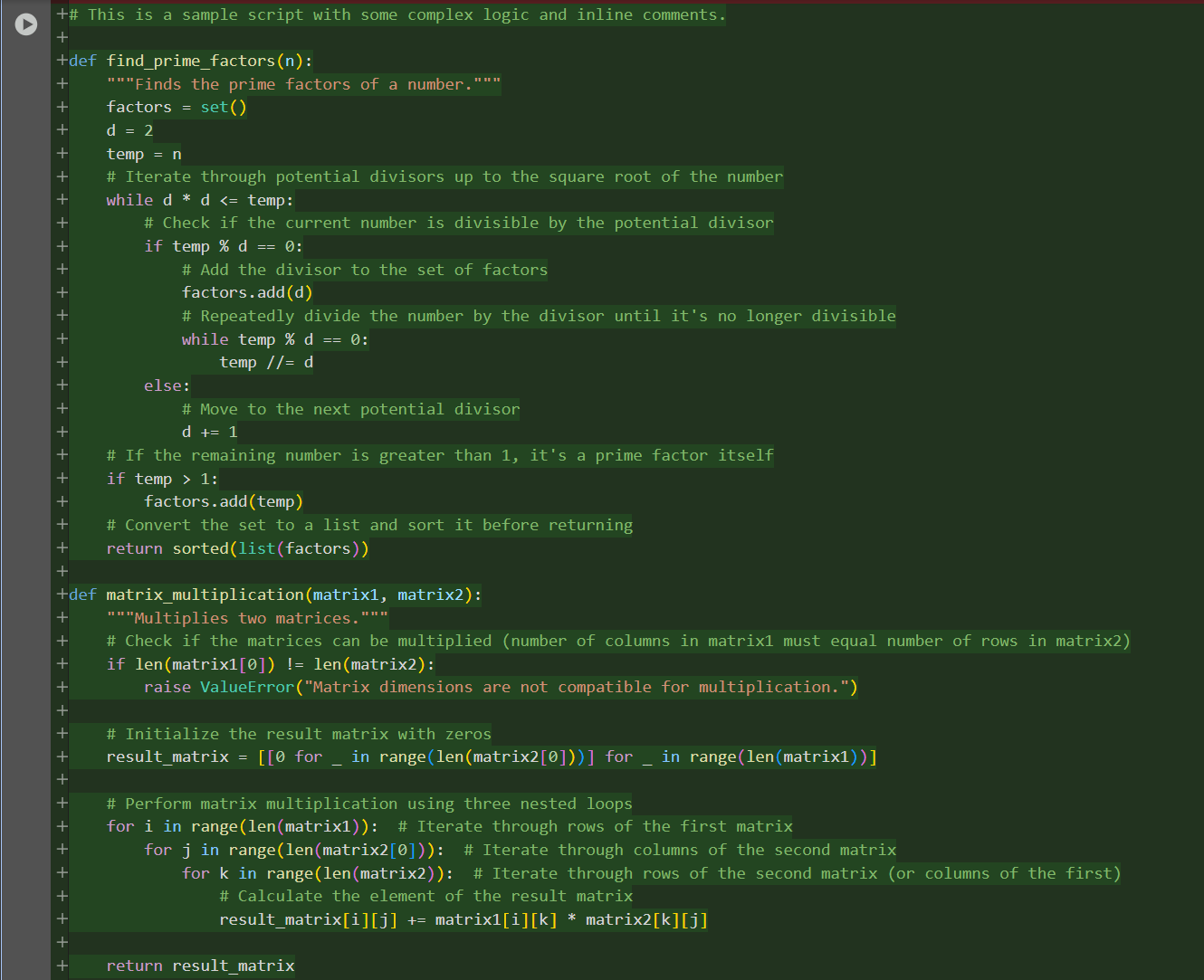


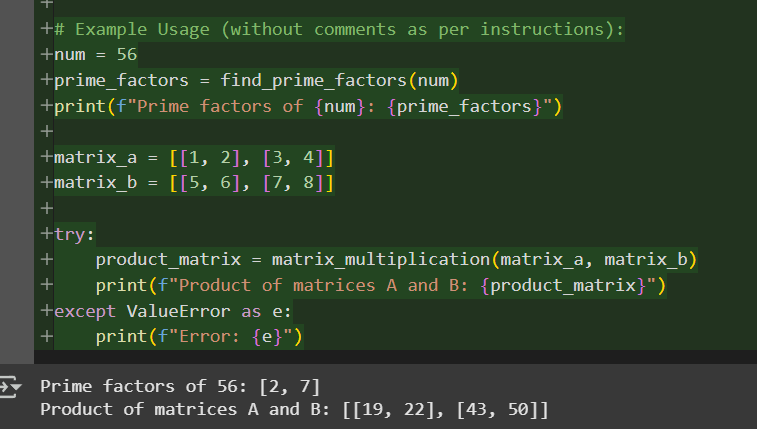


**Observation:** In this code we can Observe that the AI is Generating the different Types of codes and it .It is Having all types if Functions and Correctly formatted google style docstring

**Task Description #2 (Documentation – Inline Comments for Complex  
Logic)  
• Task: Use AI to add meaningful inline comments to a Python  
program explaining only complex logic parts.  
• Instructions:  
o Provide a Python script without comments to the AI.  
o Instruct AI to skip obvious syntax explanations and focus  
only on tricky or non-intuitive code sections.  
o Verify that comments improve code readability and  
maintainability.  
• Expected Output #2:  
o Python code with concise, context-aware inline comments  
for complex logic blocks**

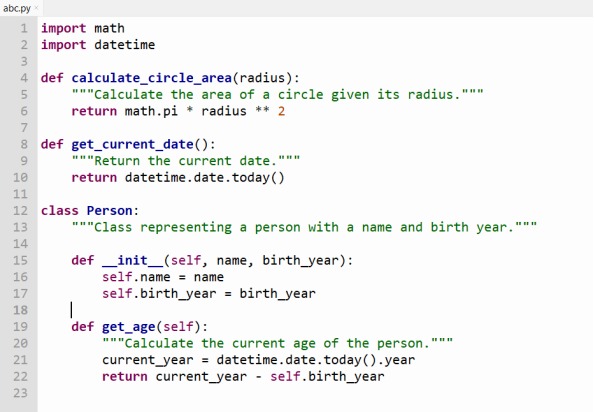
**Code and Output:**

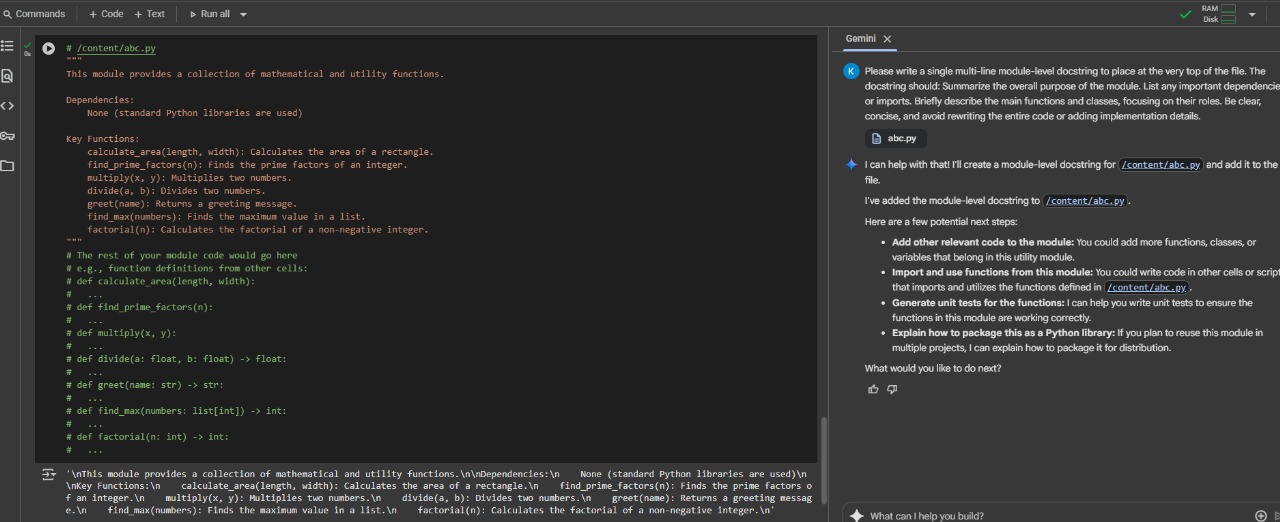
****

****

Observation:In this code we can Observe That It is Giving Python code with concise, context-aware inline comments for complex Logic Block

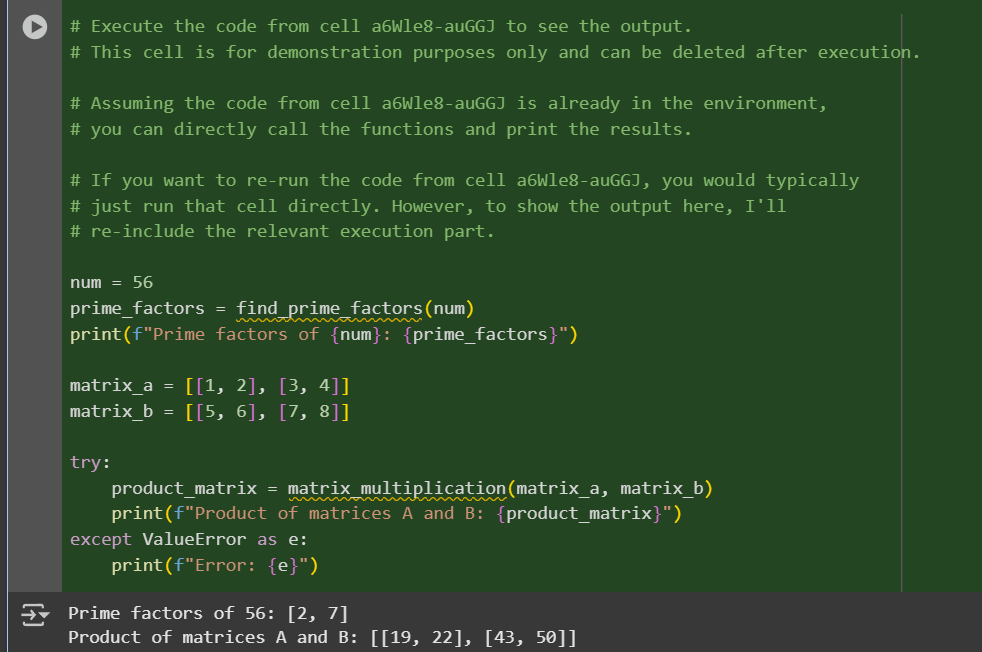
**Task Description #3 (Documentation – Module-Level Documentation)  
• Task: Use AI to create a module-level docstring summarizing the  
purpose, dependencies, and main functions/classes of a Python  
file.**

**• Instructions:  
o Supply the entire Python file to AI.  
o Instruct AI to write a single multi-line docstring at the top  
of the file.  
o Ensure the docstring clearly describes functionality and  
usage without rewriting the entire code.  
• Expected Output #3:  
o A complete, clear, and concise module-level docstring at  
the beginning of the file**.



**Task Description #4 (Documentation – Convert Comments to  
Structured Docstrings)  
• Task: Use AI to transform existing inline comments into  
structured function docstrings following Google style.  
• Instructions:  
o Provide AI with Python code containing inline comments.  
o Ask AI to move relevant details from comments into  
function docstrings.  
o Verify that the new docstrings keep the meaning intact  
while improving structure.  
• Expected Output #4:  
o Python code with comments replaced by clear,  
standardized docstrings**

Code and Output:

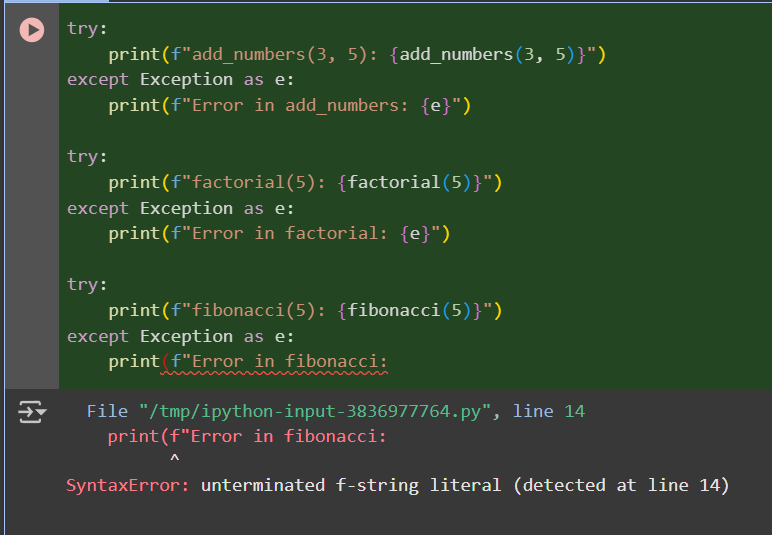
****

Observation: In this code we can see that the code is Converting inline comments into Google-style docstrings improves clarity and consistency in documentation.

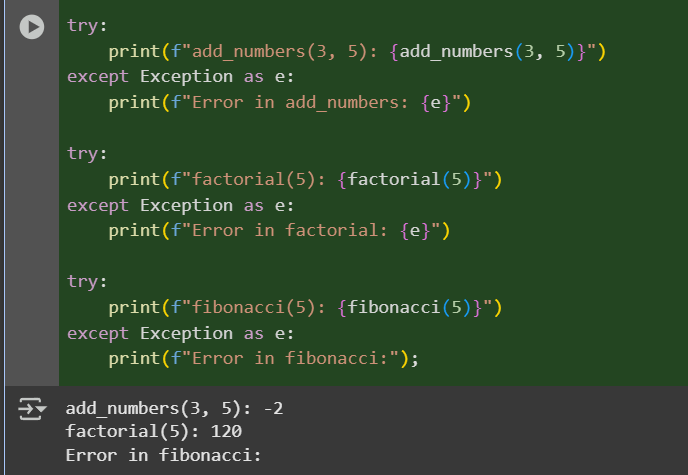
**Task Description #5 (Documentation – Review and Correct  
Docstrings)  
• Task: Use AI to identify and correct inaccuracies in existing  
docstrings.  
• Instructions:  
o Provide Python code with outdated or incorrect  
docstrings.  
o Instruct AI to rewrite each docstring to match the current  
code behavior.  
o Ensure corrections follow Google-style formatting.  
• Expected Output #5:  
o Python file with updated, accurate, and standardized  
docstrings.**

**Code and Output:**

**Before Error Correction:**

****

**After Correction:**

****

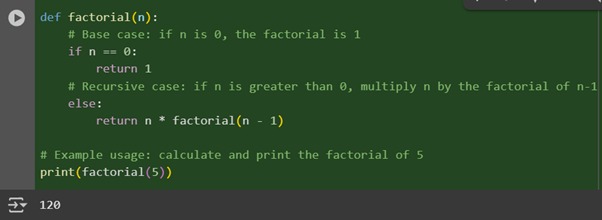
**Observation: In this code is generating the corrected code and Give the Output by correcting the code**

**Task Description #6 (Documentation – Prompt Comparison  
Experiment)**

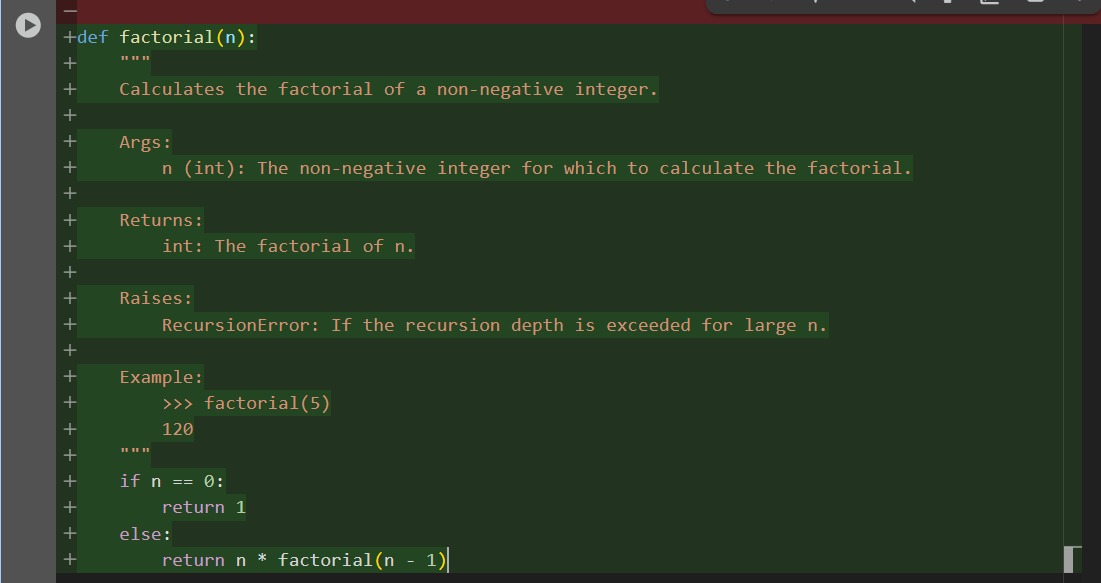
**• Task: Compare documentation output from a vague prompt and a  
detailed prompt for the same Python function.  
• Instructions:  
o Create two prompts: one simple (“Add comments to this  
function”) and one detailed (“Add Google-style docstrings  
with parameters, return types, and examples”).  
o Use AI to process the same Python function with both  
prompts.  
o Analyze and record differences in quality, accuracy, and  
completeness.  
• Expected Output #6:  
o A comparison table showing the results from both  
prompts with observations.**

**Code and Output:**

**With Comments:**



**DocString;**



**Observation:It this codes we can Observe that the Code is generating two types of codes in which we the first code Is generating the code with comments and second code we can see the code is generating code is giving code with Docstring.**