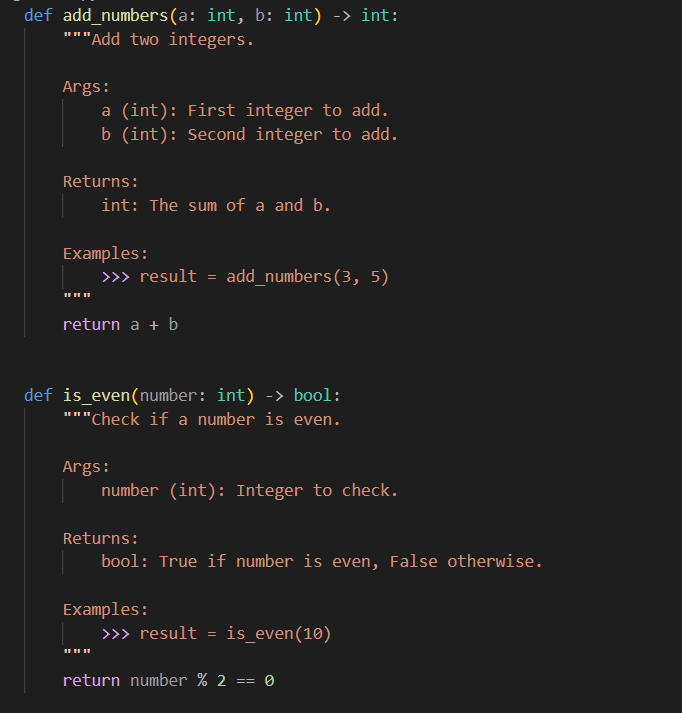
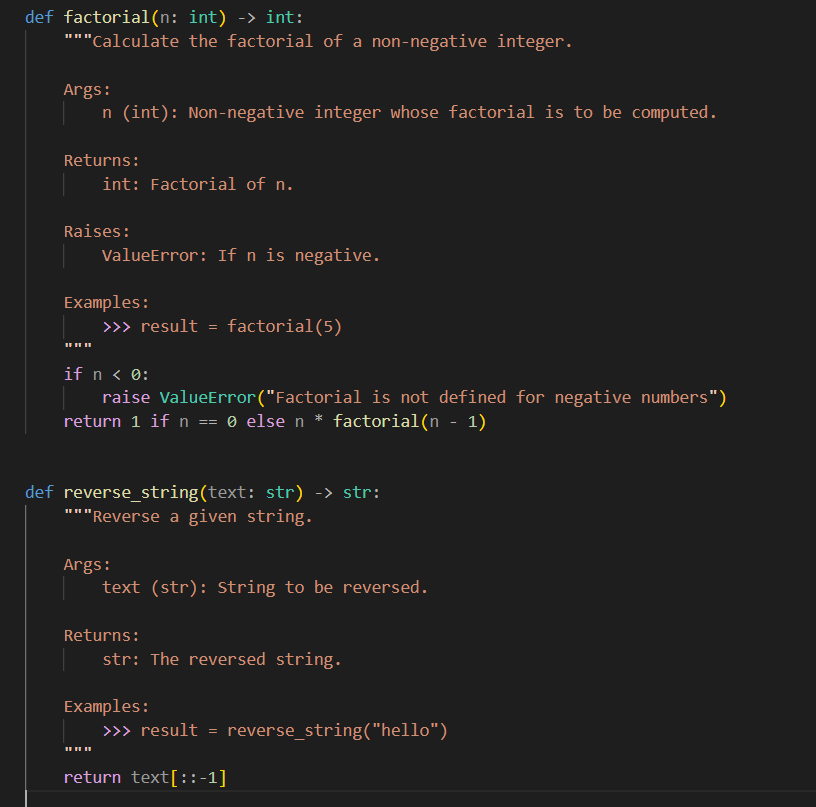
[**https://drive.google.com/file/d/1zGmLvoBwK5sT-4NL1P48QEADSYZga-PO/view?usp=sharing**](https://drive.google.com/file/d/1zGmLvoBwK5sT-4NL1P48QEADSYZga-PO/view?usp=sharing)

**Task Description #1** (Documentation – Google-Style Docstrings for Python Functions)

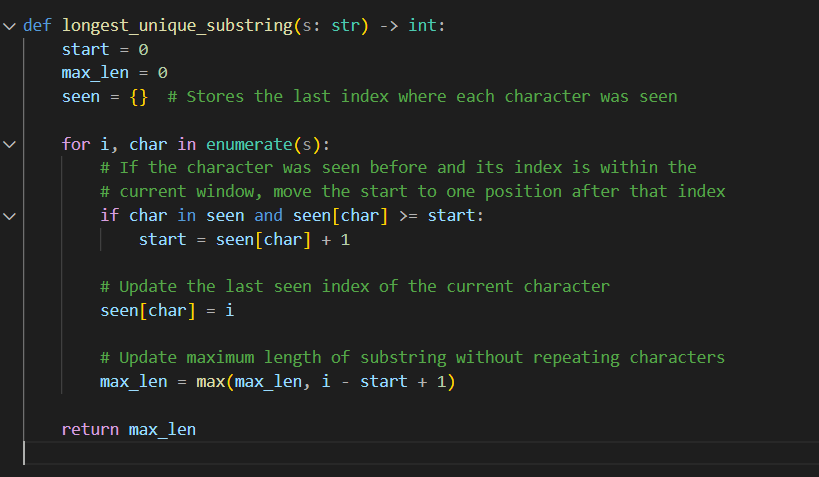
* Task: Use AI to add Google-style docstrings to all functions in a given Python script.
* Instructions:
  + Prompt AI to generate docstrings without providing any input-output examples.
  + Ensure each docstring includes:
    - Function description
    - Parameters with type hints
    - Return values with type hints
    - Example usage
  + Review the generated docstrings for accuracy and formatting.





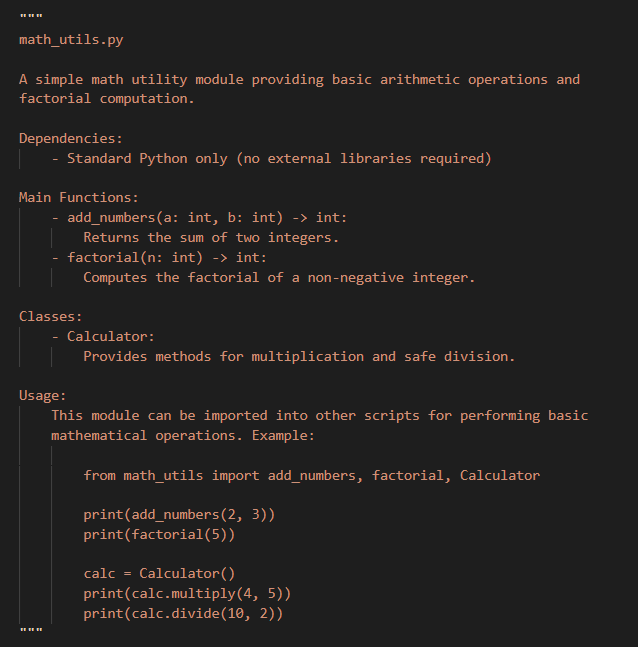
**Task Description #2** (Documentation – Inline Comments for Complex Logic)

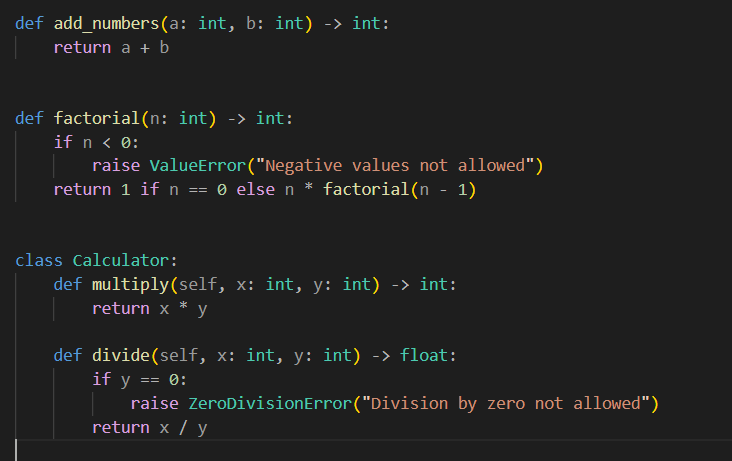
* Task: Use AI to add meaningful inline comments to a Python program explaining only complex logic parts.
* Instructions:
  + Provide a Python script without comments to the AI.
  + Instruct AI to skip obvious syntax explanations and focus only on tricky or non-intuitive code sections.
  + Verify that comments improve code readability and maintainability.



**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:
  + Supply the entire Python file to AI.
  + Instruct AI to write a single multi-line docstring at the top of the file.
  + Ensure the docstring clearly describes functionality and usage without rewriting the entire code.

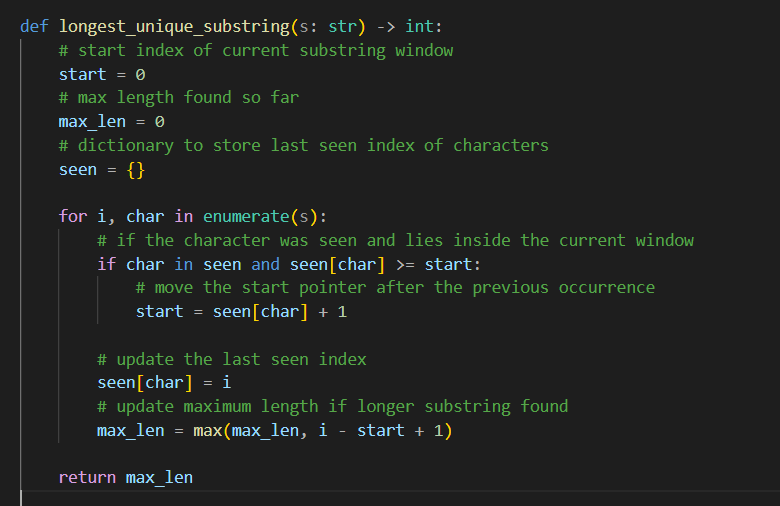




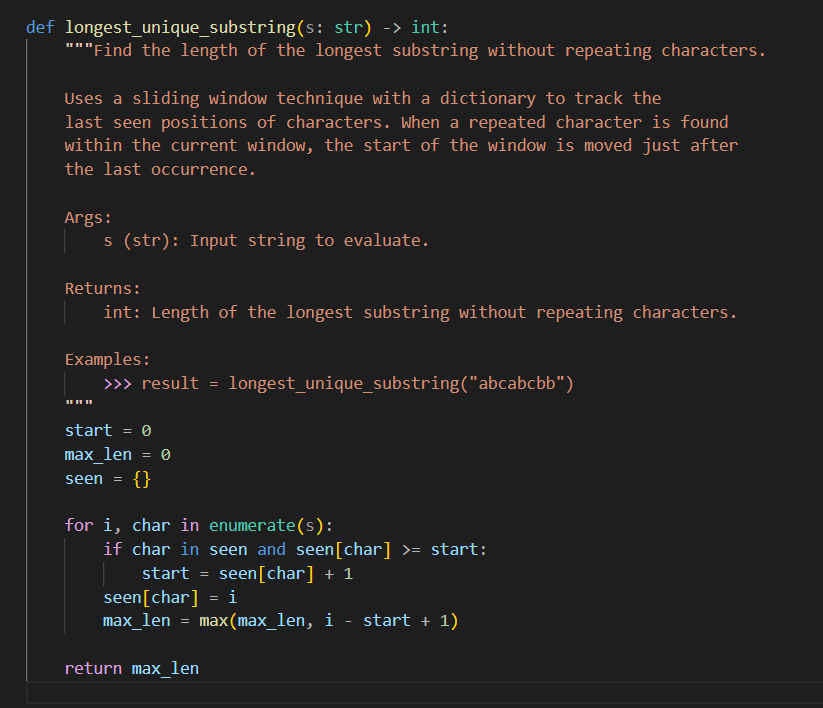
**Task Description #4** (Documentation – Convert Comments to Structured Docstrings)

* Task: Use AI to transform existing inline comments into structured function docstrings following Google style.
* Instructions:
  + Provide AI with Python code containing inline comments.
  + Ask AI to move relevant details from comments into function docstrings.
  + Verify that the new docstrings keep the meaning intact while improving structure.

Befor:



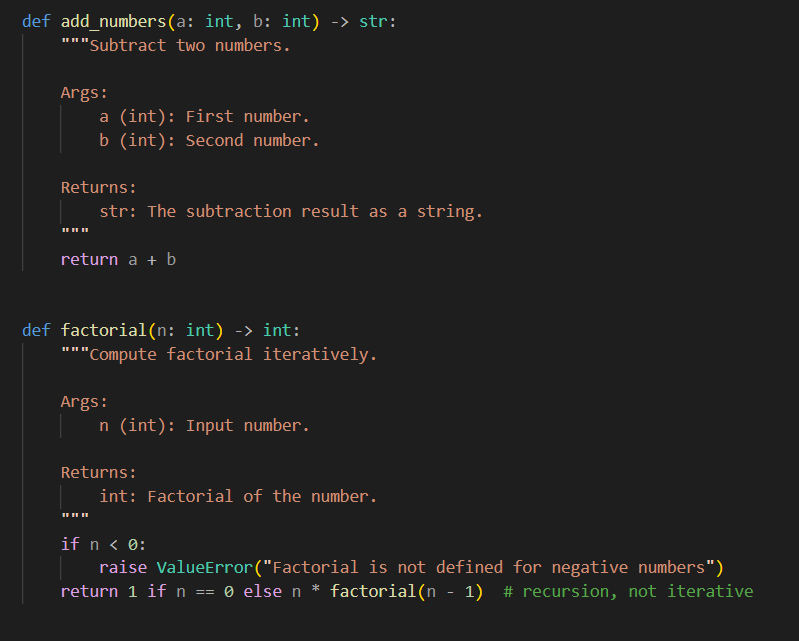
After:



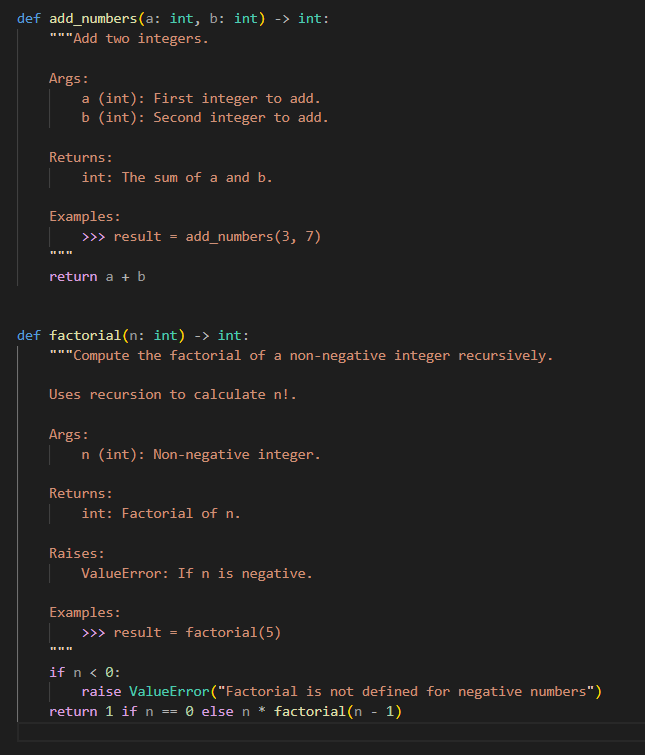
**Task Description #5** (Documentation – Review and Correct Docstrings)

* Task: Use AI to identify and correct inaccuracies in existing docstrings.
* Instructions:
  + Provide Python code with outdated or incorrect docstrings.
  + Instruct AI to rewrite each docstring to match the current code behavior.
  + Ensure corrections follow Google-style formatting.

Before:



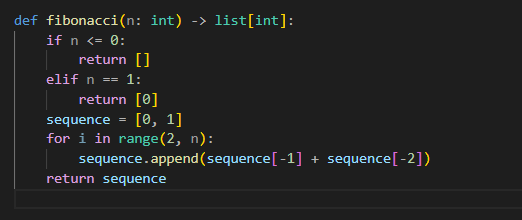
After:



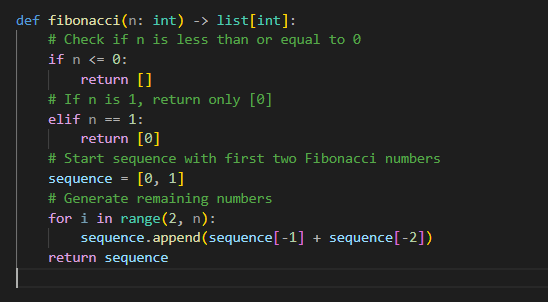
**Task Description #6** (Documentation – Prompt Comparison Experiment)

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

Before:



**Prompt:** *“Add comments to this function.”*



**Prompt:** *“Add Google-style docstrings with parameters, return types, and examples.”*

