HINO:2403A51286  
 Assignment:9.2

**Task#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.

**Prompt:**

# Add Google-style docstrings to all functions in the following Python script.

# Requirements:

# - Each docstring must include:

# \* A short function description

# \* Parameters with type hints

# \* Return values with type hints

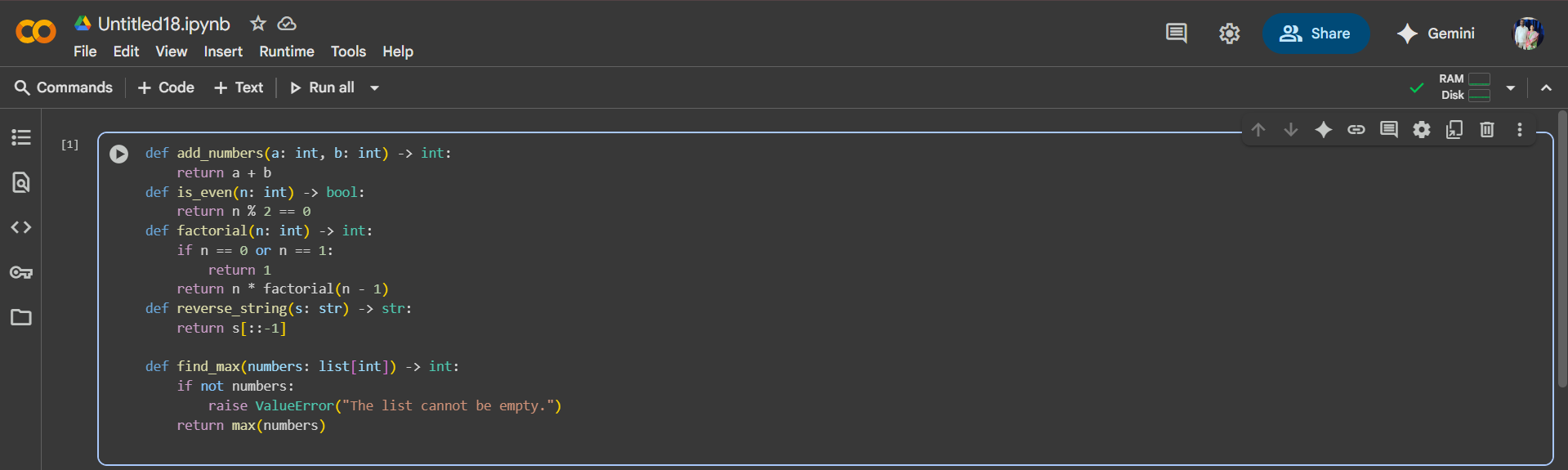
# \* Example usage (but do not provide actual input/output values)

# - Do not include input/output examples with real numbers, just the format.

# - Follow proper Google-style formatting conventions.

# - Ensure accuracy and clarity in the generated docstrings.

**Code:**

****

**Task#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.

**Prompt:**

(Documentation – Inline Comments for Complex Logic)

• Task: Add inline comments to the provided Python program.

• Instructions:

- Only explain tricky or non-intuitive logic (e.g., sliding window movement, majority vote algorithm).

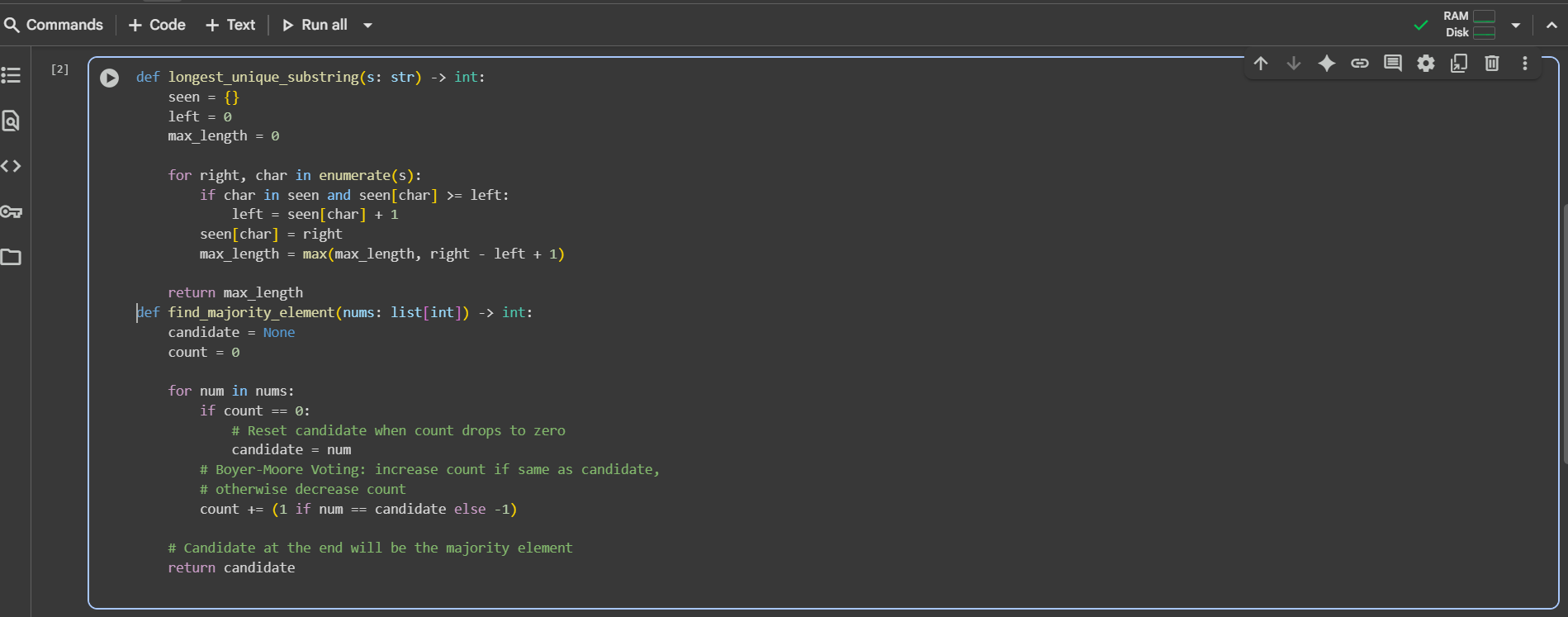
- Skip obvious syntax or simple steps like variable assignments or return statements.

- Ensure comments improve readability and maintainability for someone new to the logic.

• Expected Output:

- Python code with concise inline comments explaining complex parts.

**Code:**

****

**Task#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.

**Prompt:**

(Documentation – Module-Level Documentation)

• Task: Create a single module-level docstring for the provided Python file.

• Instructions:

- Summarize the purpose of the module in 3–6 sentences.

- List any dependencies (imported modules/libraries).

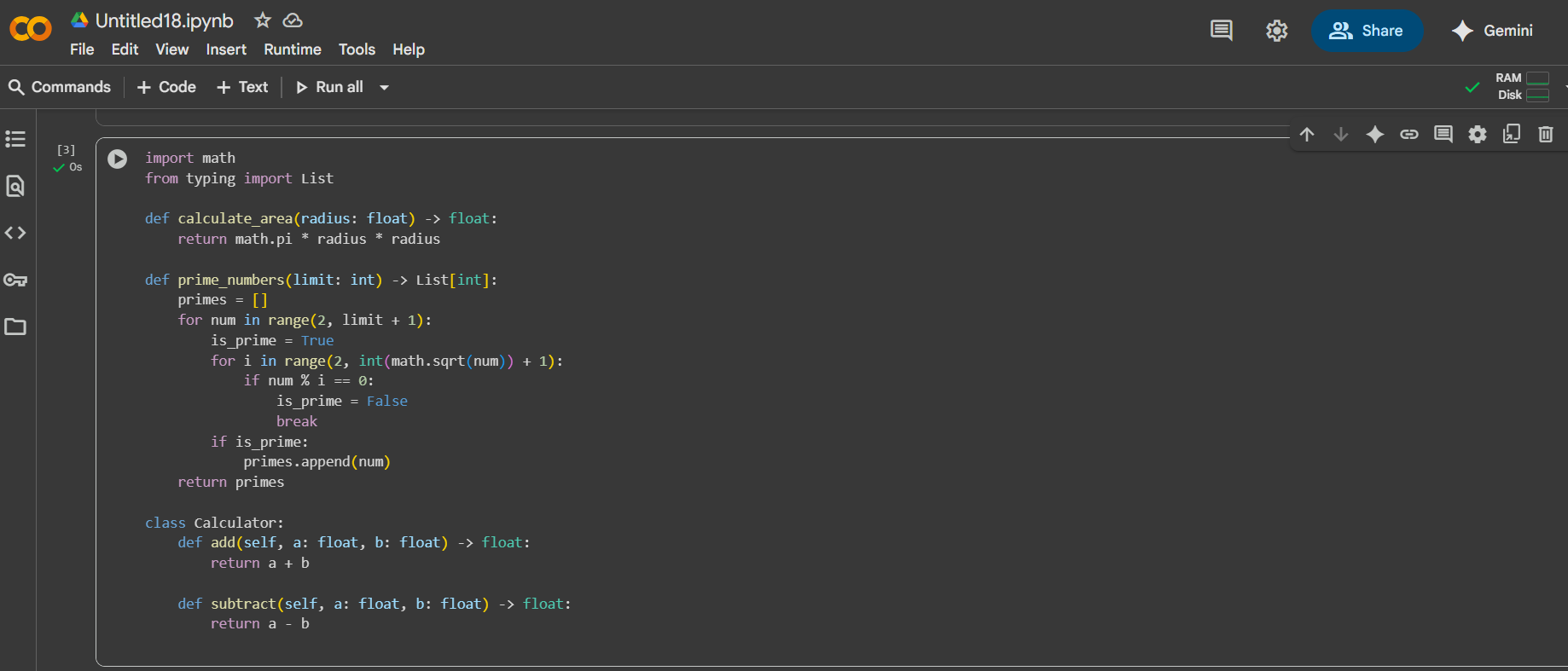
- Briefly describe the main functions and classes included.

- Do not rewrite or duplicate the full code.

• Expected Output:

- A clear and concise multi-line docstring at the very top of the file.

**Code:**

****

**Task#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

**Prompt:**

(Documentation – Convert Comments to Structured Docstrings)

• Task: Transform the inline comments in the provided Python code into structured Google-style docstrings.

• Instructions:

- Move relevant details from inline comments into function docstrings.

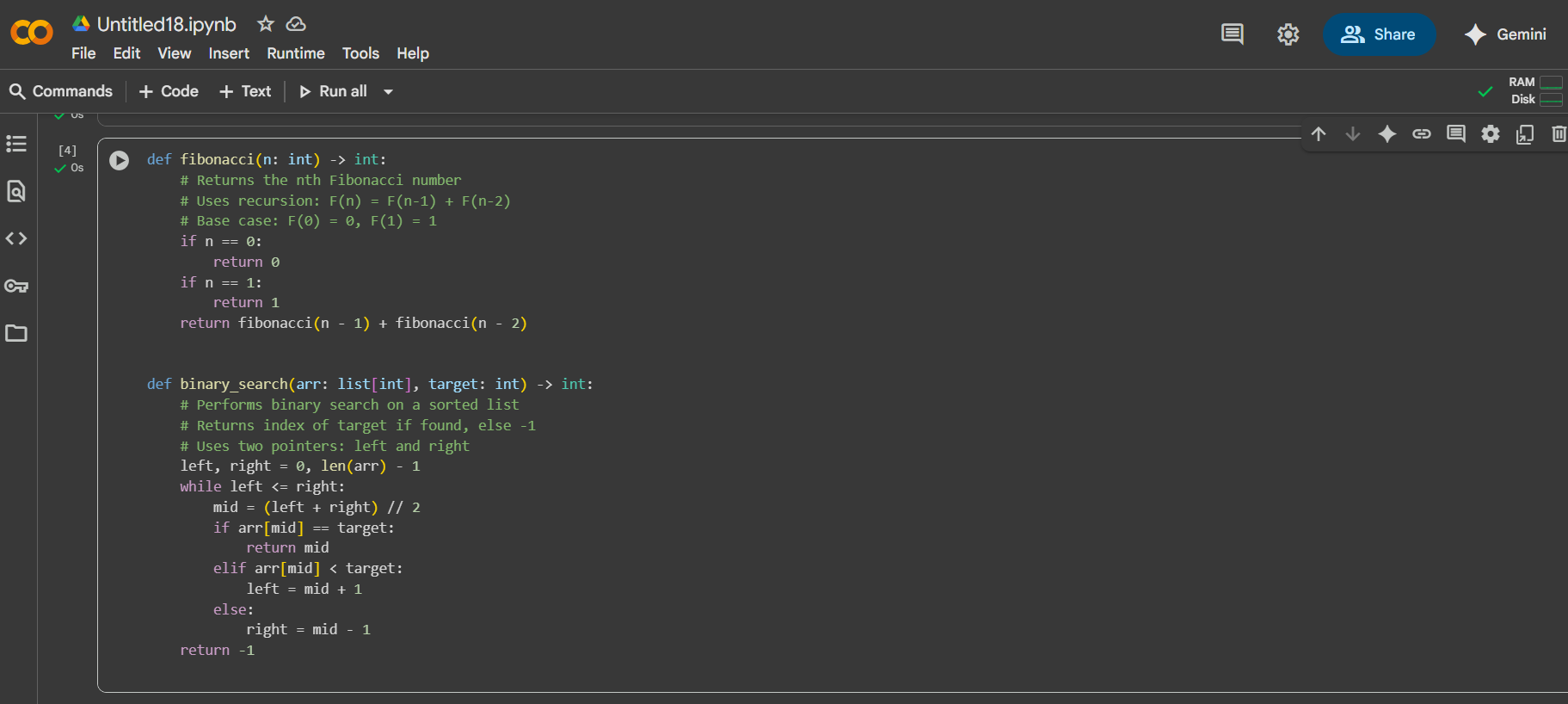
- Follow Google-style format with description, Args, Returns, and Example.

- Remove redundant inline comments after converting them into docstrings.

• Expected Output:

- Python code with standardized Google-style docstrings and no leftover inline comments.

**Code:**

****

**Task#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  
docstring

**Prompt:**

(Documentation – Review and Correct Docstrings)

• Task: Identify and correct inaccuracies in the provided Python code docstrings.

• Instructions:

- Ensure each docstring accurately describes the function's purpose and behavior.

- Rewrite them using Google-style docstring format.

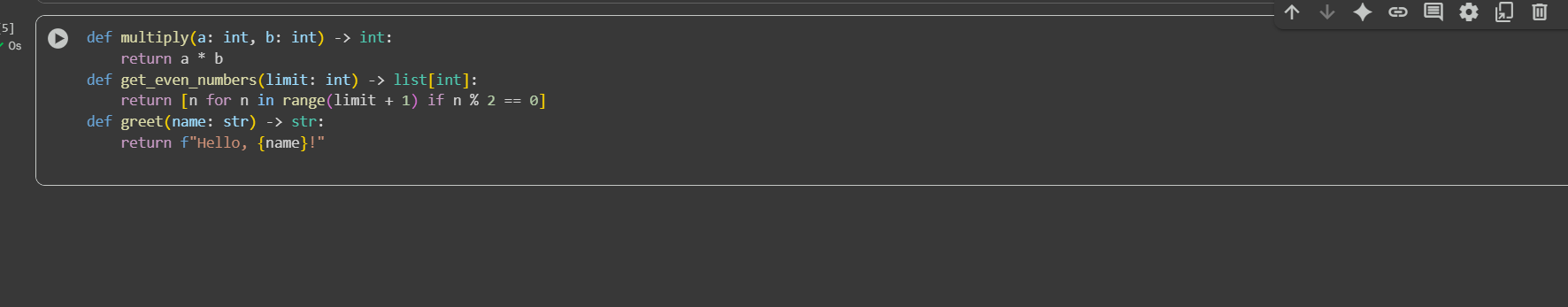
- Include Args, Returns, and Example sections where applicable.

• Expected Output:

- Python code with corrected and standardized docstrings.

**Code:**

**Example Python Code (With Incorrect Docstrings)**

****

**Task#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.

**Prompt:**

(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.

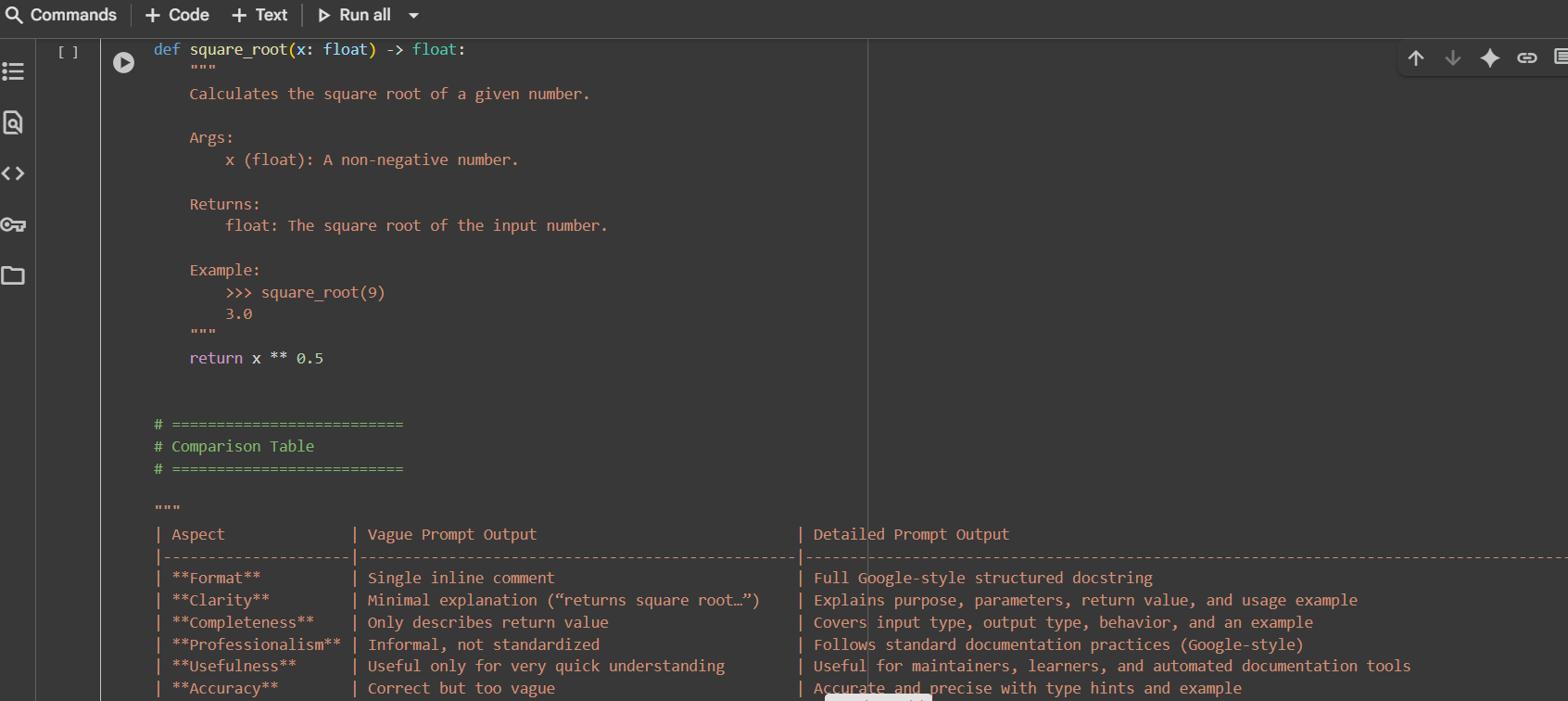
- Analyze and record differences in quality, accuracy, and completeness.

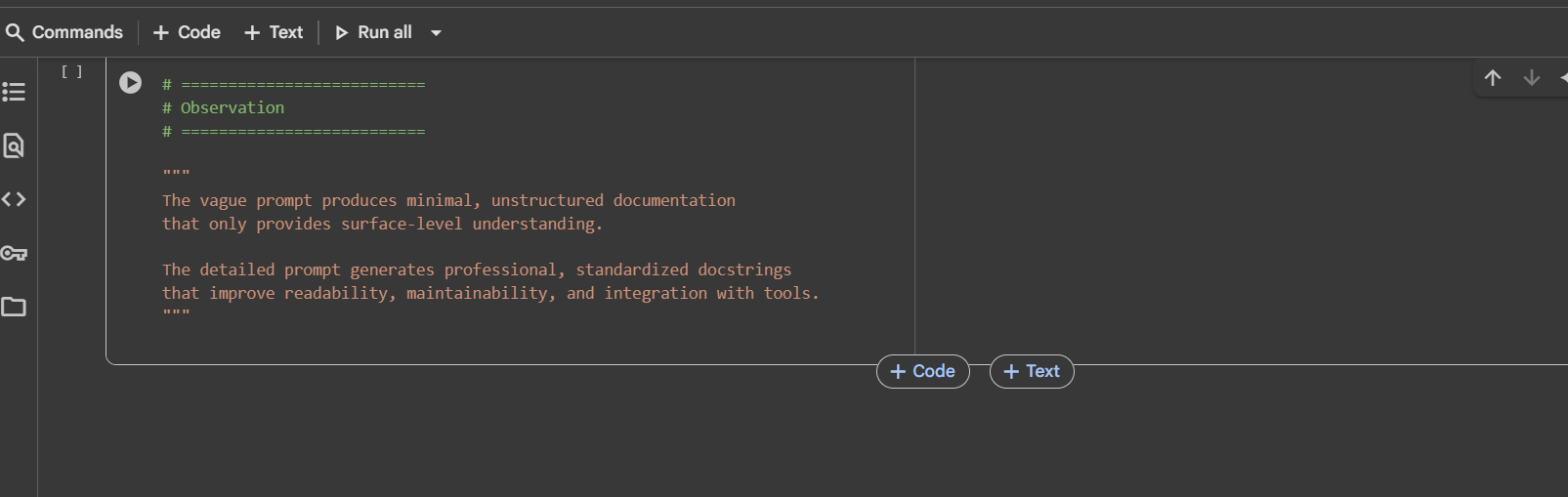
• Expected Output:

- A comparison table showing the results from both prompts with observations.

**Code:**

****

****

****