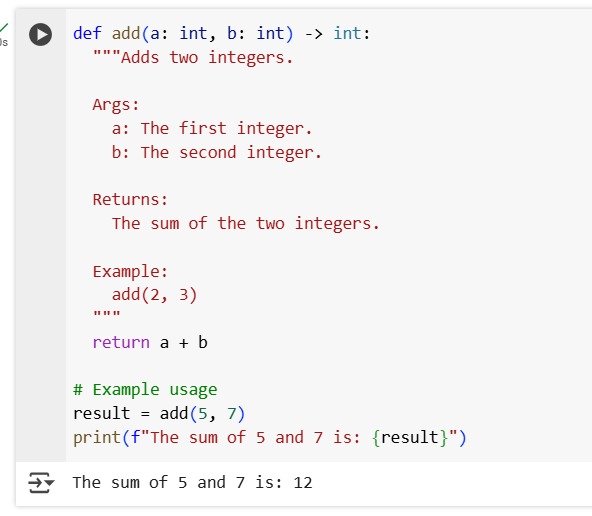
**ASSIGNMENT-9.1**

NAME:PRANAVI PAMERA

Roll no:2403a51328

Batch:13

**TASK-1:**

* Task: Use AI to add Google-style docstrings to all functions in a given Python script.
  + 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-2:**

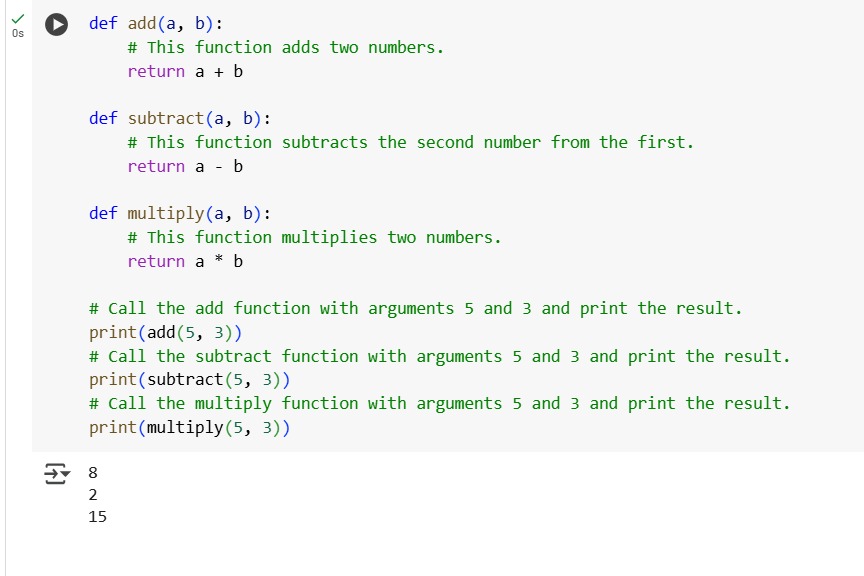
* Task: Use AI to add meaningful inline comments to a Python program explaining only complex logic parts.
* Instructions:
  + Provide a Python script without comments to the AI.
  + Instruct AI to skip obvious syntax explanations and focus only on tricky or non-intuitive code sections.

Verify that comments improve code readability and maintainability.

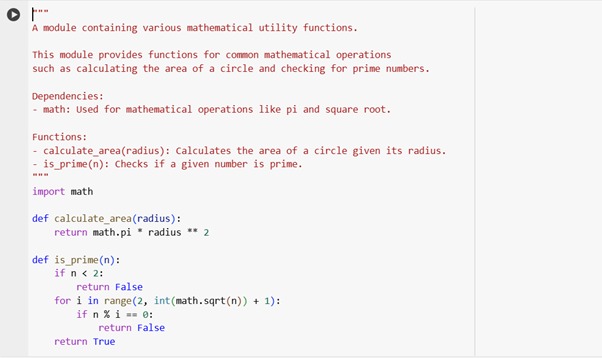
**o HERE GIVEN CODE WITHOUT INLINE COMMENTS:**

def add(a, b): return a + b def subtract(a, b): return a - b def multiply(a, b): return a \* b print(add(5, 3)) print(subtract(5, 3)) print(multiply(5, 3))

**WITH INLINE COMMENTS:**

**TASK-3:**

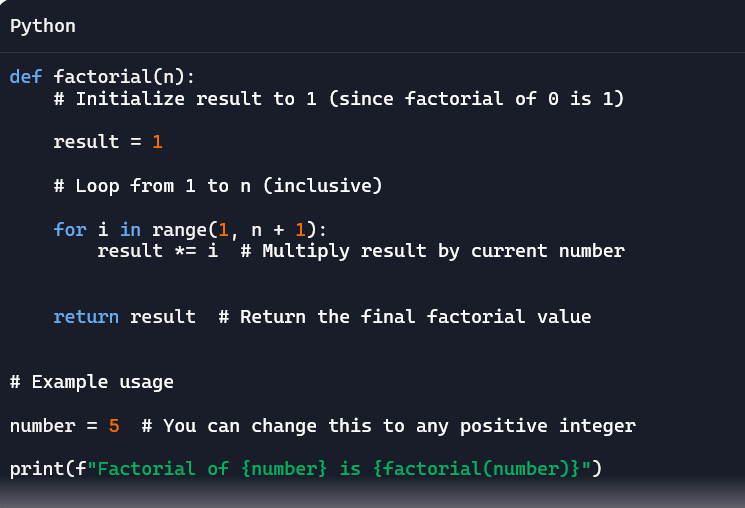
* Task: Use AI to create a module-level docstring summarizing the purpose, dependencies, and main functions/classes of a Python file.
* Instructions:
  + Supply the entire Python file to AI.
  + Instruct AI to write a single multi-line docstring at the top of the file.
  + Ensure the docstring clearly describes functionality and usage without rewriting the entire code.



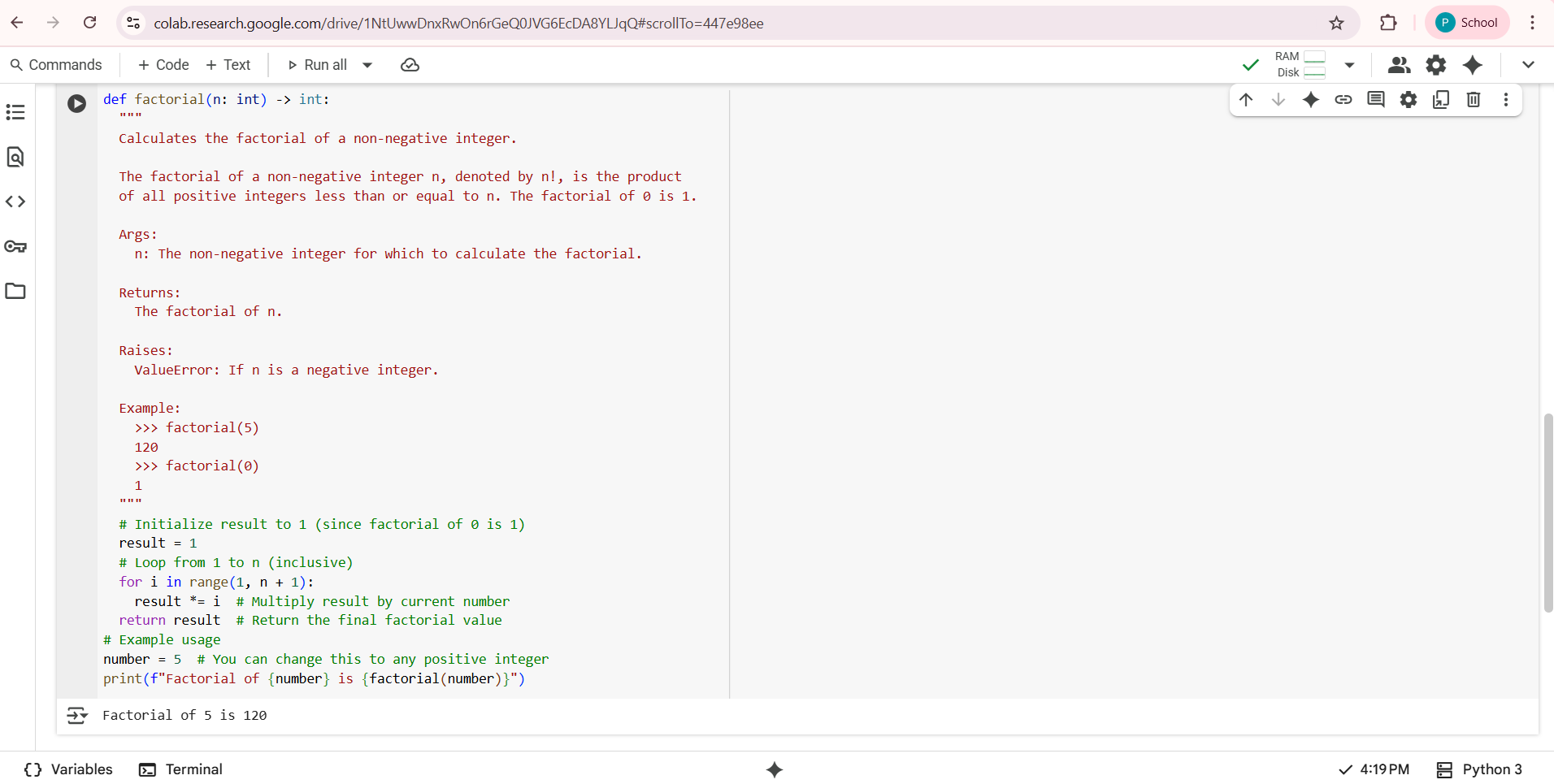
**TASK-4:**

* 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



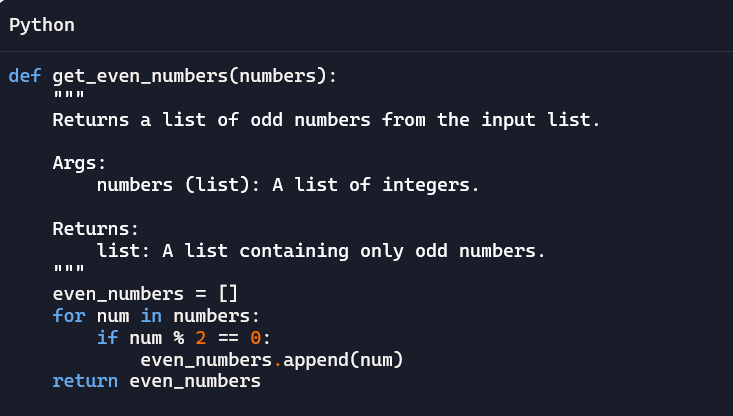
**Python function with inline comments converted into docstrings**



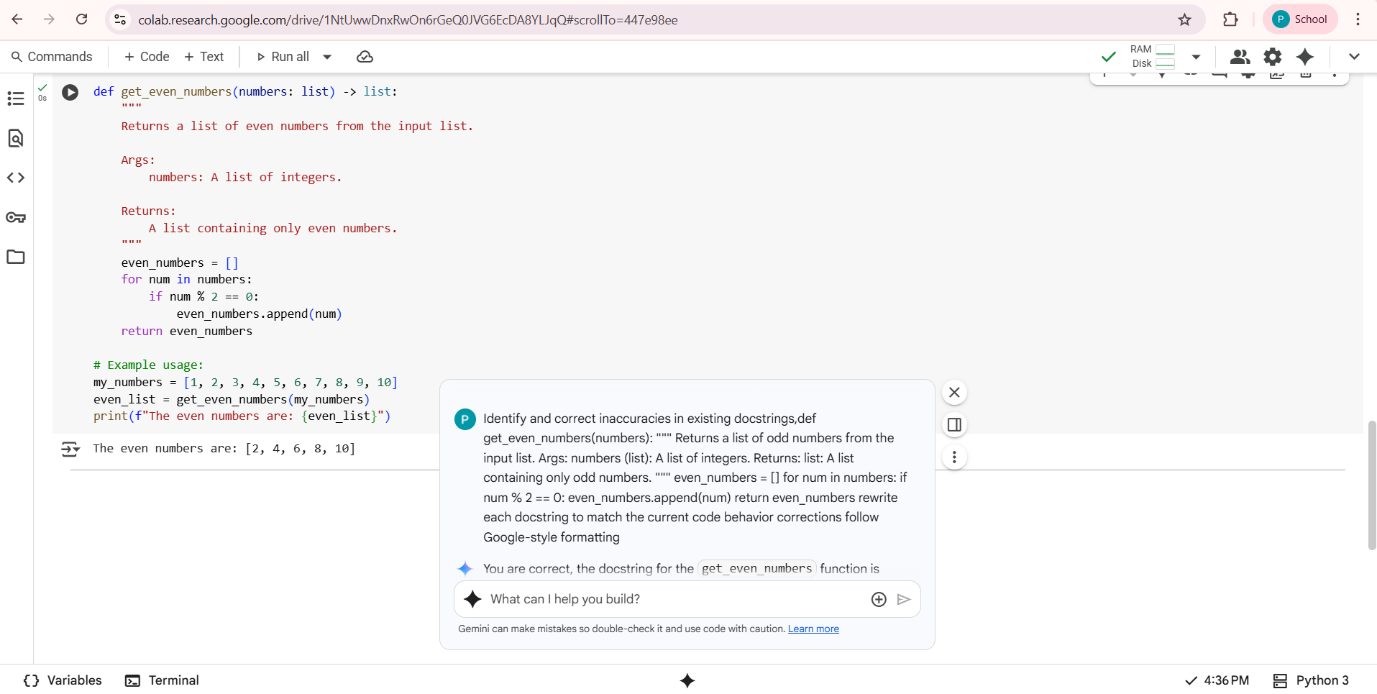
**TASK-5:**

* 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.

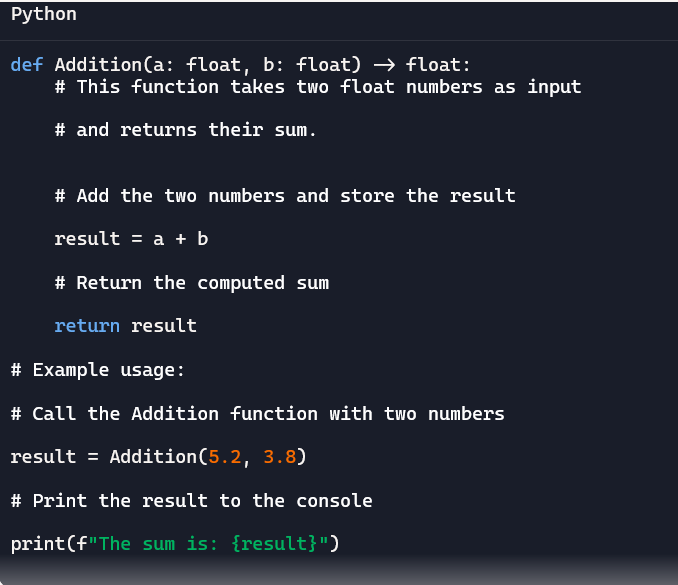
Ensure corrections follow Google-style formatting

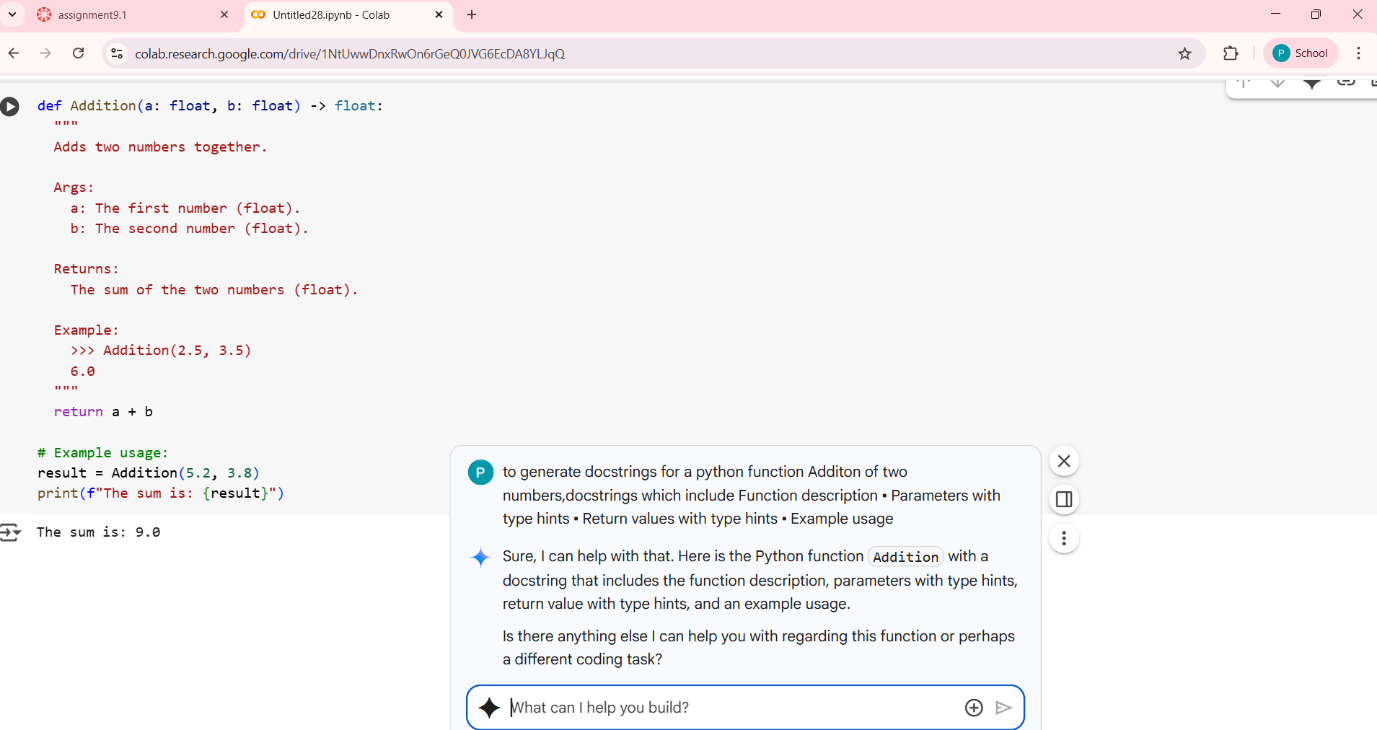
****

**CORRECTING THE ABOVE CODE:**

****

**TASK-6:**

* 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.
* 



**Comparisionndetailed prompt and vague prompt:**

| **Feature** | **Generated Docstring** | **Inline Comments** |
| --- | --- | --- |
| **Quality** | High (Standardized format, parsable by tools) | Lower (Dispersed, not standardized) |
| **Accuracy** | High (Describes purpose, args, return, example) | High (Describes specific code steps) |
| **Completeness** | High (Includes description, args, return, example) | Lower (Explains steps, lacks overall function overview) |
| **Purpose** | Formal function/API documentation | Explaining specific code logic to readers of the code |