ASSIGNMENT : 9.2

H.N.T.O : 2403A51292

BATCH : 12

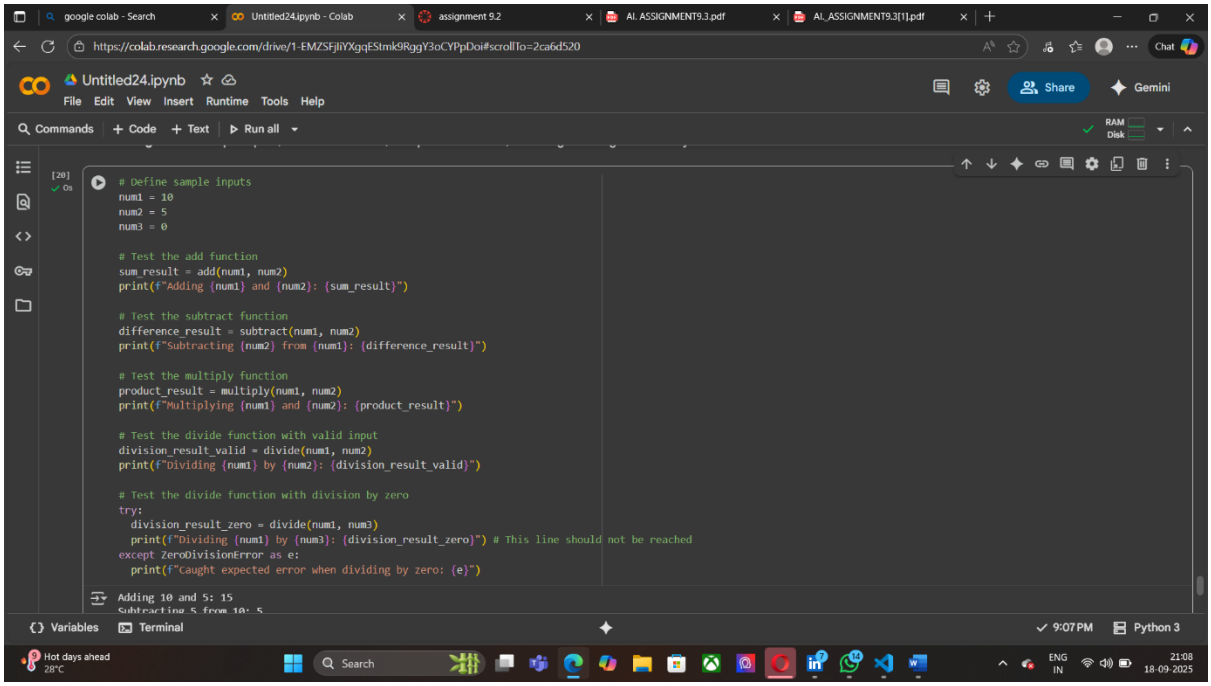
Task Description 1 :

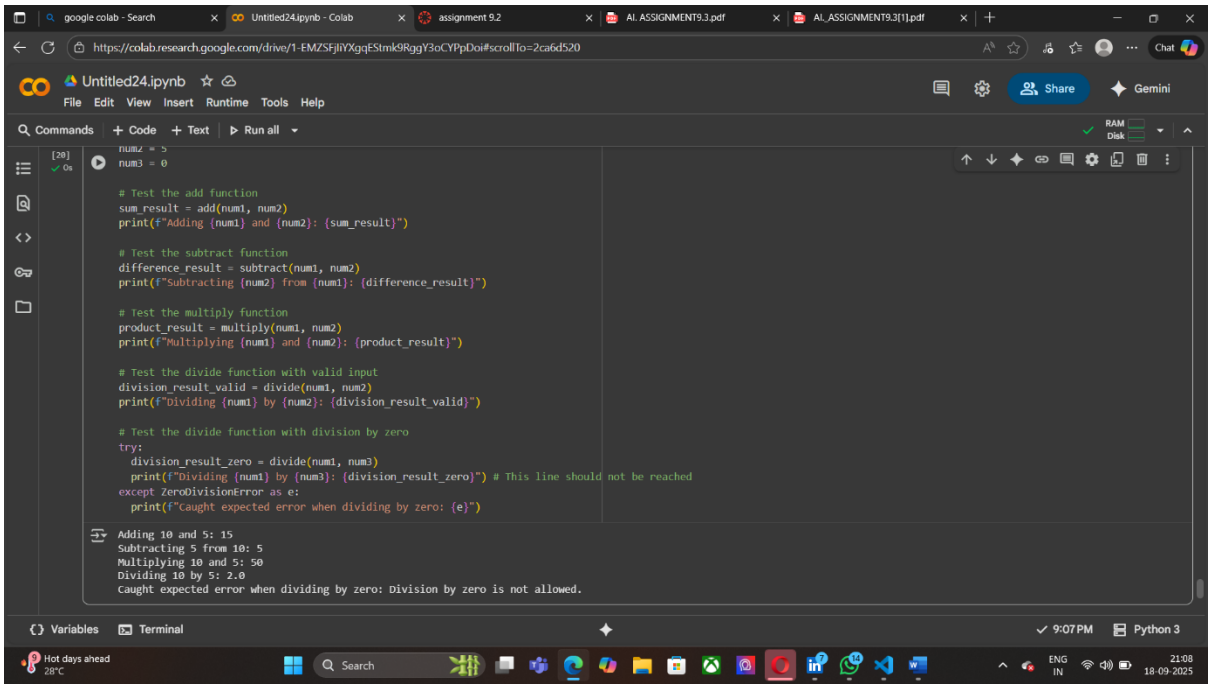
(Documentation – Google-Style Docstrings for
Python Functions)
• Task: Use AI to add Google-style docstrings to all functions in a
given Python script.
• Instructions:
o Prompt AI to generate docstrings without providing any
input-output examples.
o Ensure each docstring includes:
▪ Function description
▪ 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.

Prompt:

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

- Do not include any input-output examples.
- Each docstring should include:
  - Function description
  - Parameters with type hints
  - Return values with type hints
  - Example usage
- Ensure the docstrings are accurate and properly formatted.





Task Description 2:

(Documentation – Inline Comments for Complex Logic)

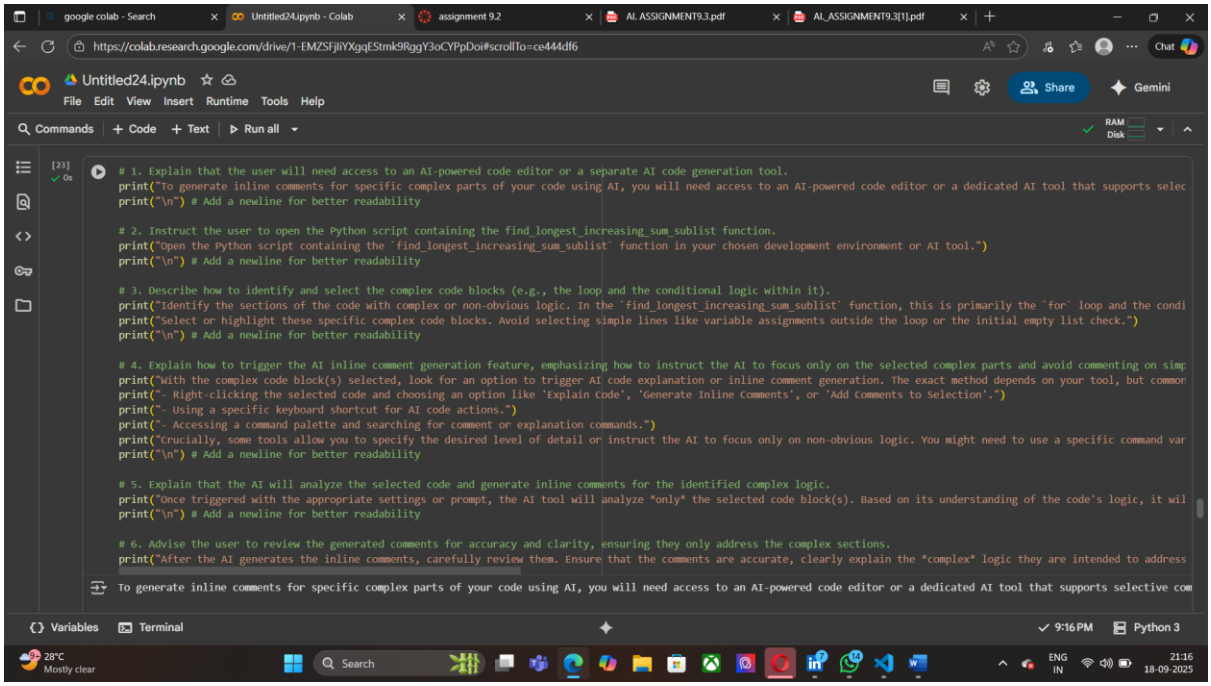
• Task: Use AI to add meaningful inline comments to a Python program explaining only complex logic parts.

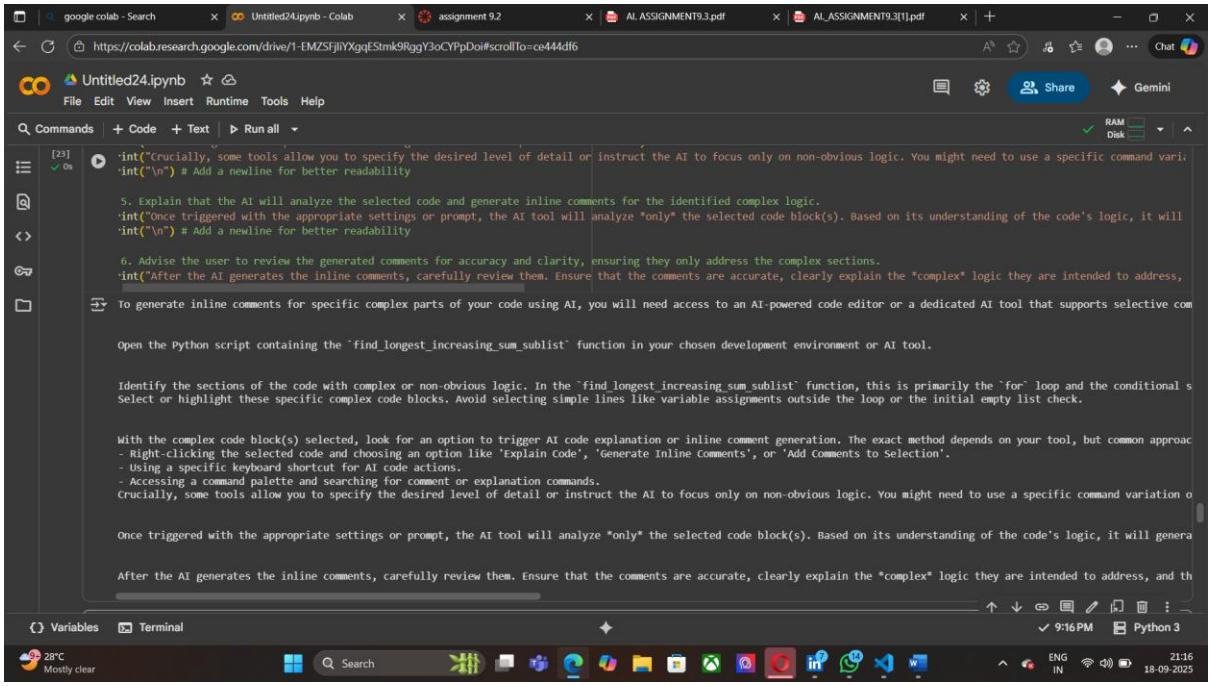
• Instructions:
o Provide a Python script without comments to the AI.
o Instruct AI to skip obvious syntax explanations and focus only on tricky or non-intuitive code sections.
o Verify that comments improve code readability and maintainability.
• Expected Output #2:
o Python code with concise, context-aware inline comments for complex logic blocks.

Prompt:

Add meaningful inline comments to the following Python program, explaining only the complex or non-intuitive logic parts.

- Skip obvious syntax explanations.
- Focus on tricky or non-obvious code sections.
- Ensure 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:

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:

Write a single multi-line module-level docstring at the top of the following Python file.

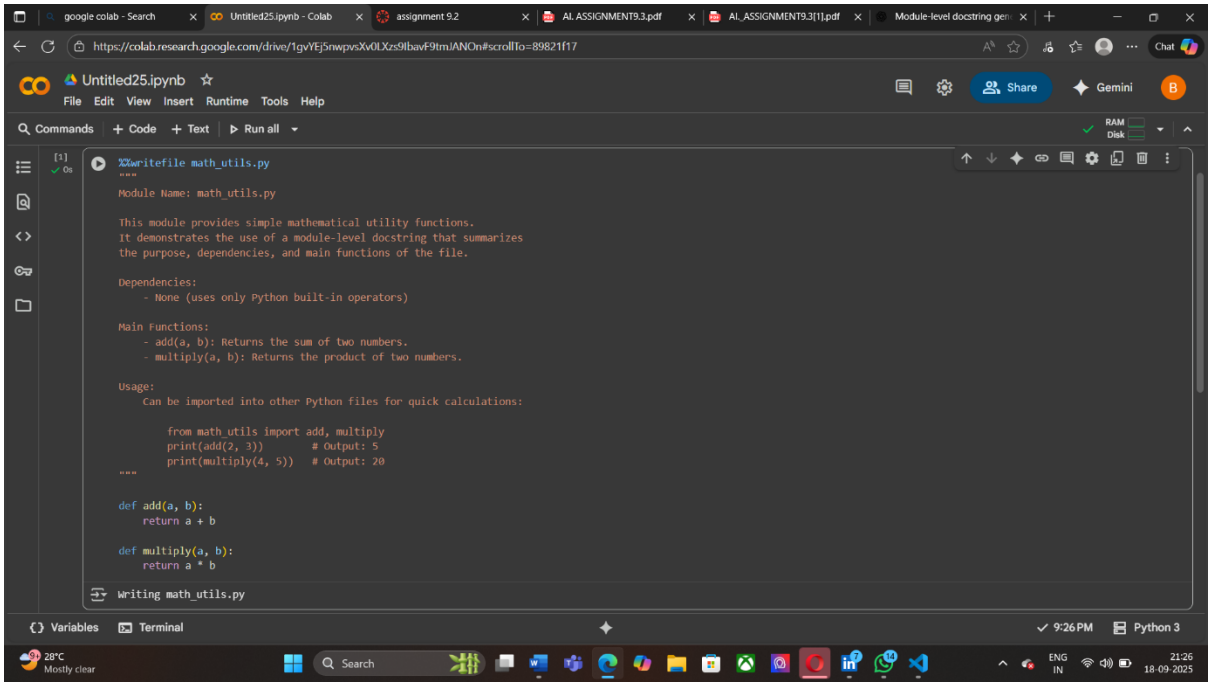
- Summarize the file's purpose, dependencies, and main functions/classes.

- Clearly describe functionality and usage without rewriting the entire code

Prompt:

Transform all existing inline comments in the following Python code into structured function docstrings using Google style.

- Move relevant details from comments into the function docstrings.
- Ensure the new docstrings keep the original meaning and improve structure.



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.

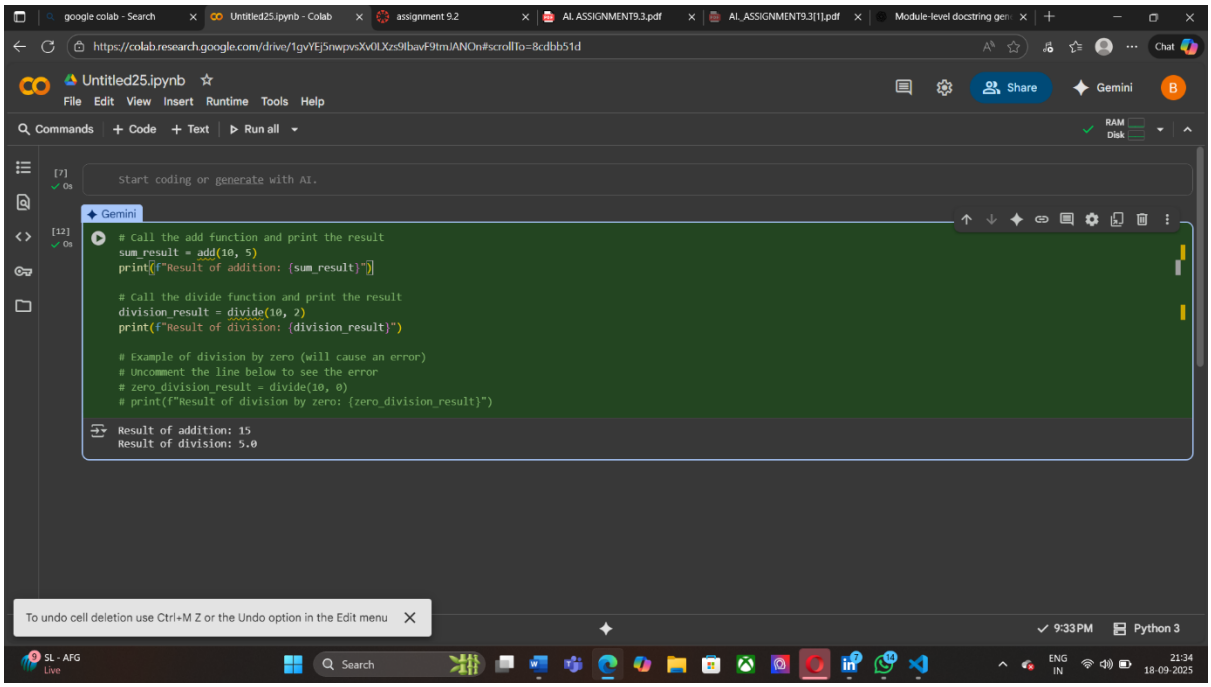
• Expected Output #4:

o Python code with comments replaced by clear, standardized docstrings.

Prompt:

Transform all existing inline comments in the following Python code into structured function docstrings using Google style.

- Move relevant details from comments into the function docstrings.
- Ensure the new docstrings keep the original meaning and improve structure.



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.

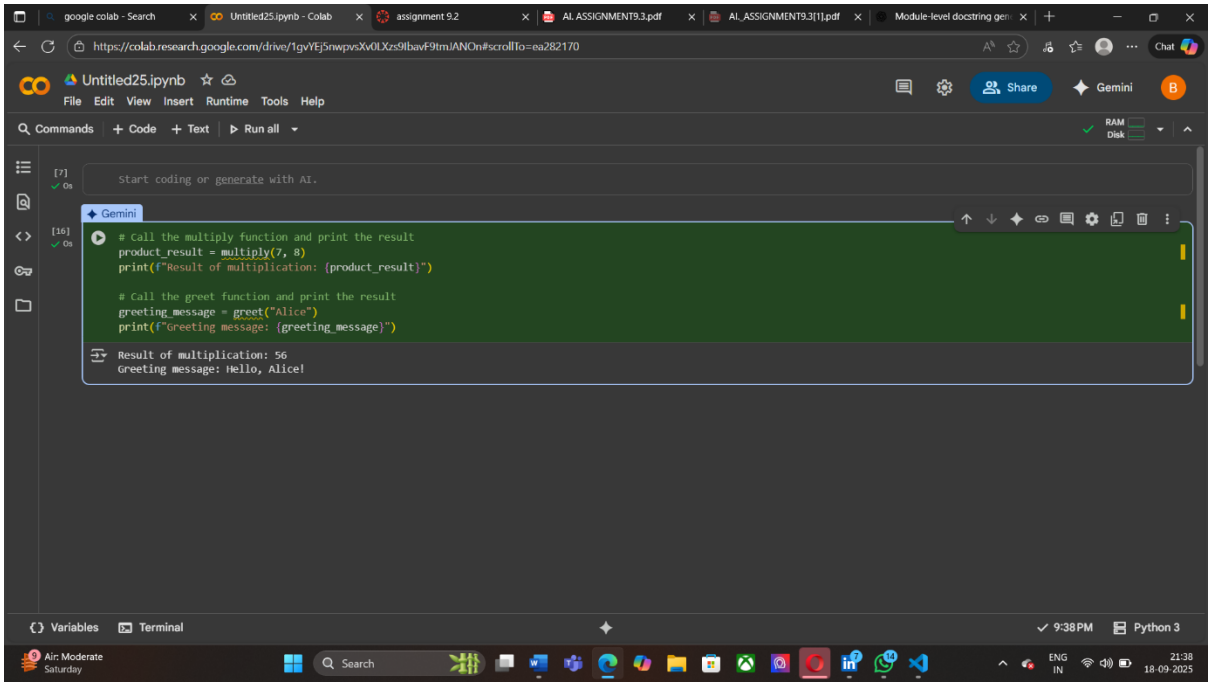
• Expected Output #5:

o Python file with updated, accurate, and standardized docstrings.

Prompt:

Review the following Python code and identify any outdated or incorrect docstrings.

- Rewrite each docstring to accurately match the current code behavior.
- Ensure all docstrings follow Google-style formatting.



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

Prompt:

Compare documentation output from a vague prompt and a detailed prompt for the same Python function:

- Vague prompt: "Add comments to this function."

- Detailed prompt: “Add Google-style docstrings with parameters, return types, and examples.”
- Use AI to process the same function with both prompts.
- Analyze and record differences in quality, accuracy, and completeness in a comparison table with observations.