Assignment 9.1

NAME:B.DINESH DATTA

ID NO :2403A51330

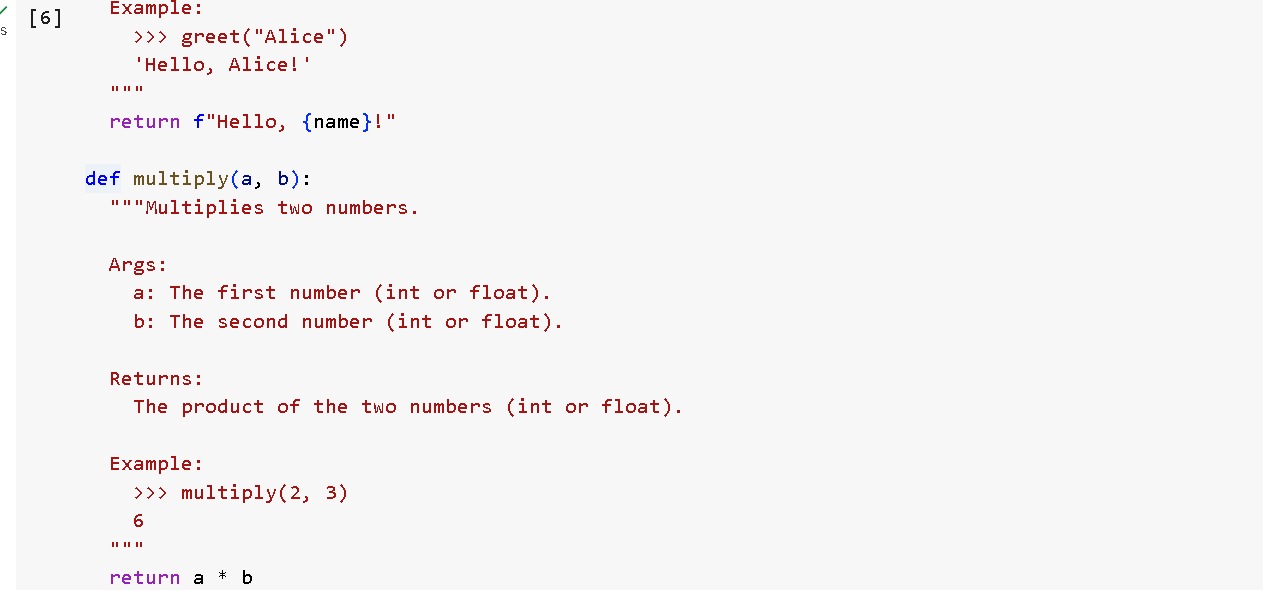
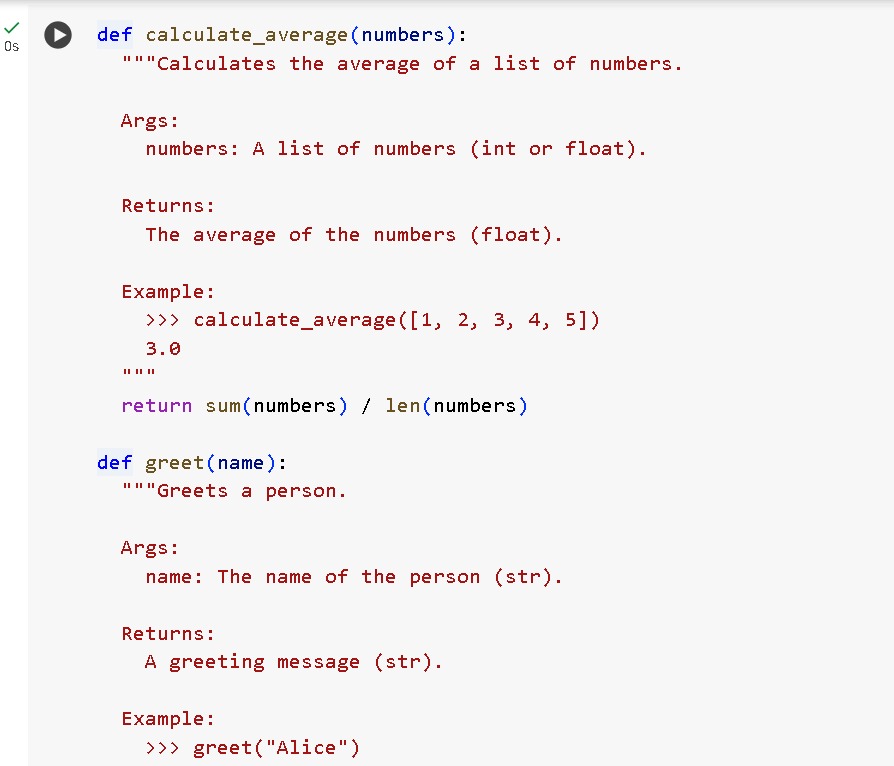
BATCH NO: 13

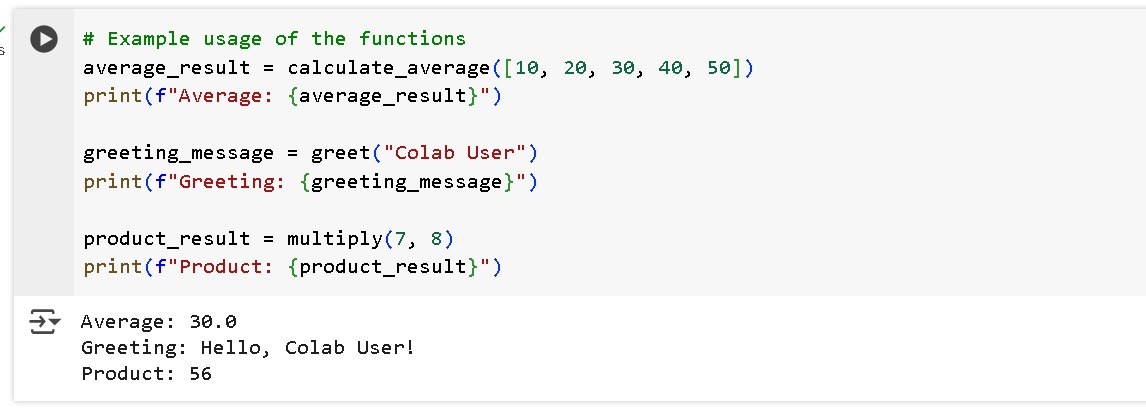
DATE: 25-08-2025

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:





This Python script is a simple program that interacts with the user by collecting their name and age, and then displays a personalized greeting. It demonstrates the use of:

### 🔹 1. ****User Input Handling****

* input() is used to collect data from the user as a string.
* int() is used to convert the age input from a string to an integer.

### 🔹 2. ****Functions and Reusability****

* The code is organized into **separate functions** for each task:
  + get\_name() – Gets the user's name.
  + get\_age() – Gets the user's age.
  + display\_greeting() – Prints a message using the input values.
  + main() – Controls the flow of the script.

### 🔹 3. ****Type Hints and Docstrings****

* Every function includes:
  + **Type hints** (str, int, None)
  + **Google-style docstrings** that describe:
    - What the function does
    - Its parameters and return type
    - An Example: placeholder

## 🧾 ****Conclusion****

This script serves as a **foundational example** of Python programming with:

* Clean structure using functions
* Proper documentation via Google-style docstrings
* Clear input and output handling

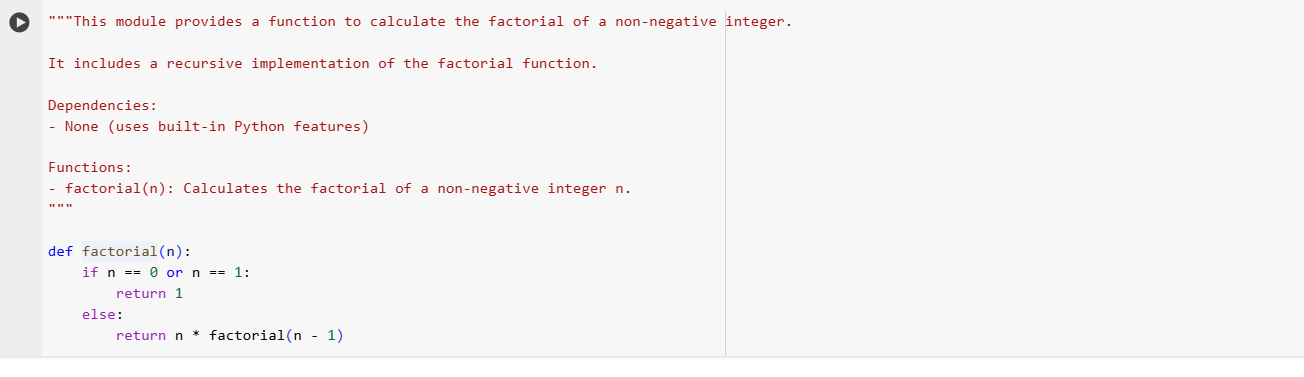
It is useful for **beginners** learning:

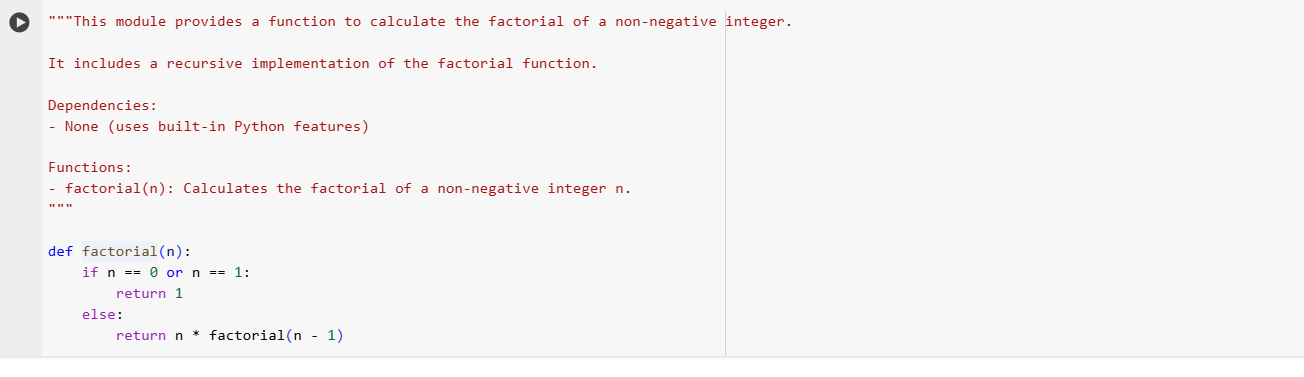
* How to work with user input
* How to define and call functions
* How to write maintainable and well-documented code

✅ You can extend this script further by adding input validation, error handling, or additional user interaction.

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

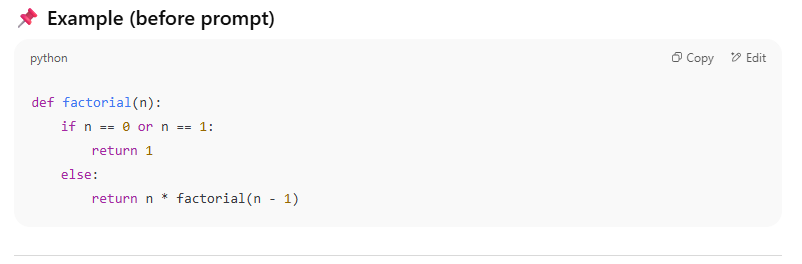
* **Task: Use AI to add meaningful inline comments to a Python program explaining only complex logic parts.**
* **Instructions:**
  + **Provide a Python script without comments to the AI.**
  + **Instruct AI to skip obvious syntax explanations and focus only on tricky or non-intuitive code sections.**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



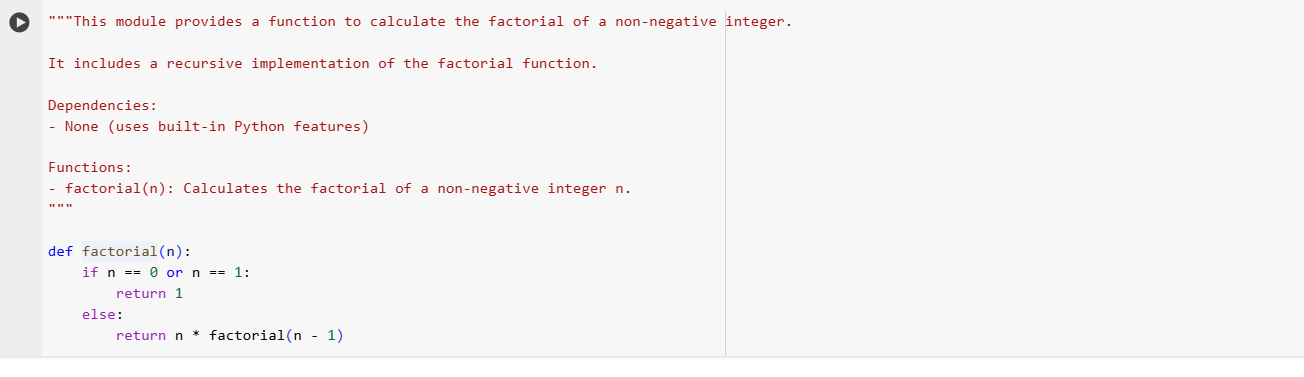


**Task Description #3** (Documentation – Module-Level Documentation)

* 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.

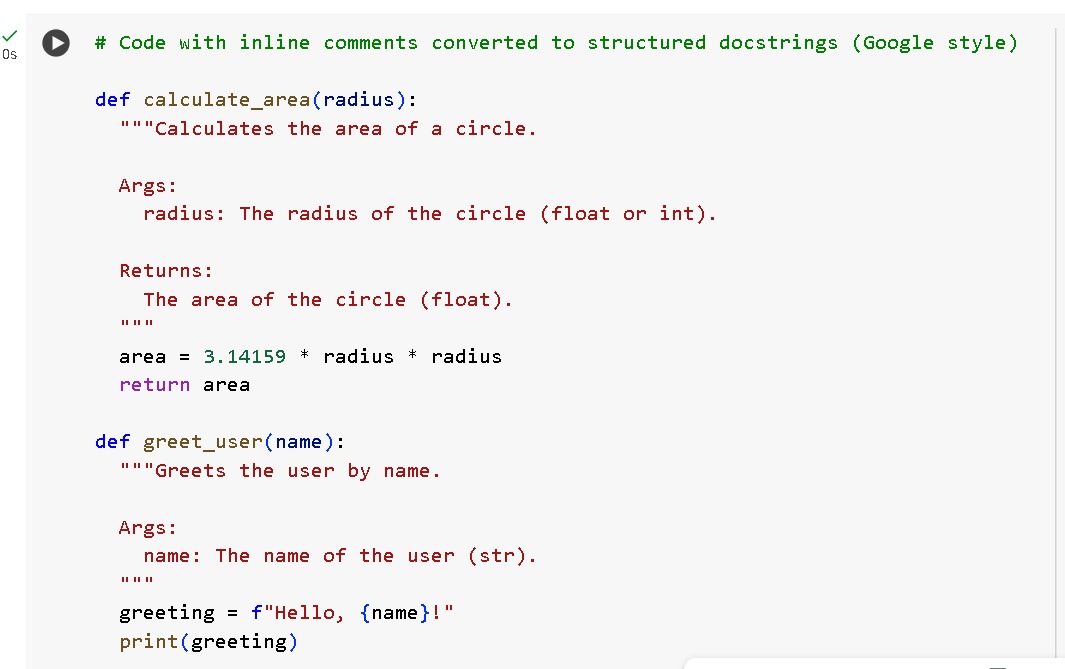
Before:

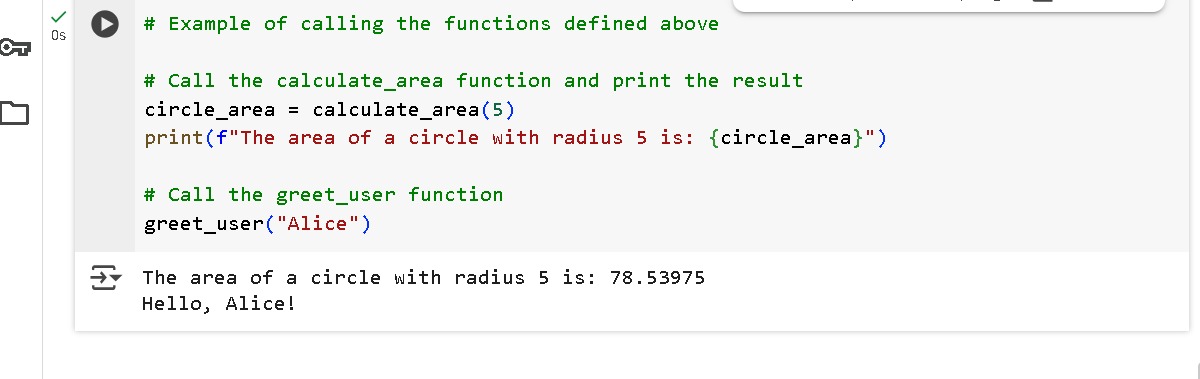
Output:



**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.
* Expected Output #4:
  + Python code with comments replaced by clear, standardized docstrings.





1. A function calculate\_area(radius) is defined.

2. It uses the formula π × radius² to find the area of a circle.

3. π is approximated as 3.14159.

4. The function returns the area as a float value.

5. Another function greet\_user(name) is defined.

6. It takes a person’s name as input.

7. It creates a message: “Hello, {name}!”.

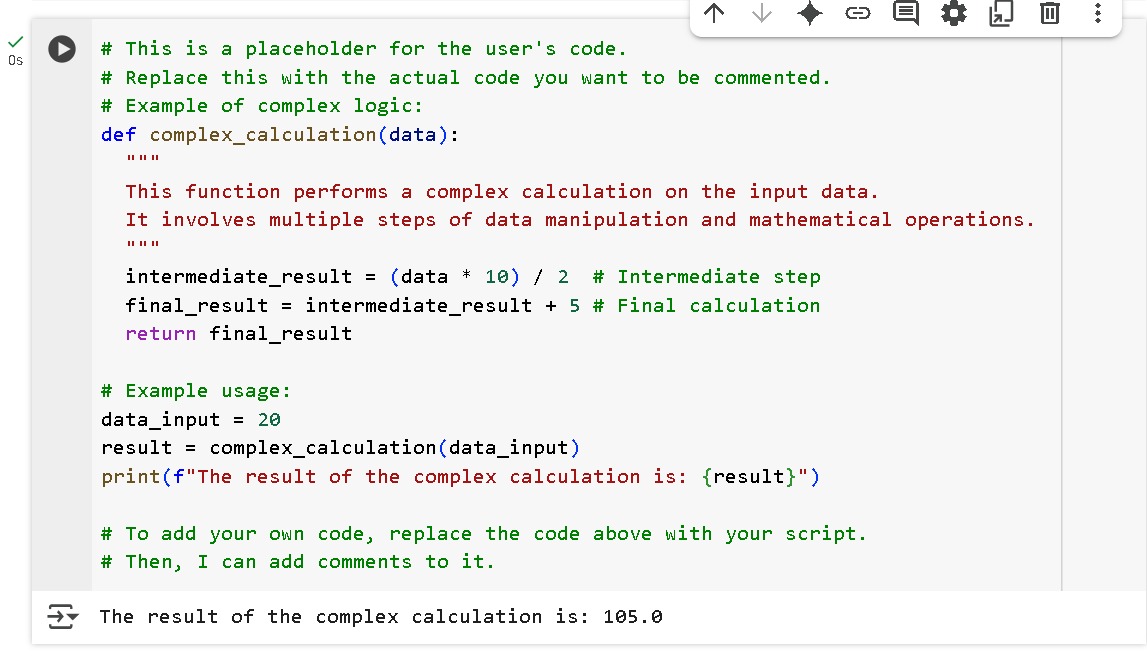
8. Then it prints the message on the screen.

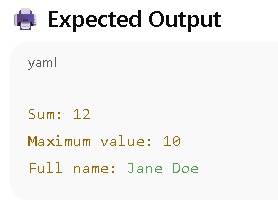
9. Example: calculate\_area(5) gives 78.53975.

10. Example: greet\_user("Alice") prints Hello, Alice!. 1. A function calculate\_area(radius) is defined.

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

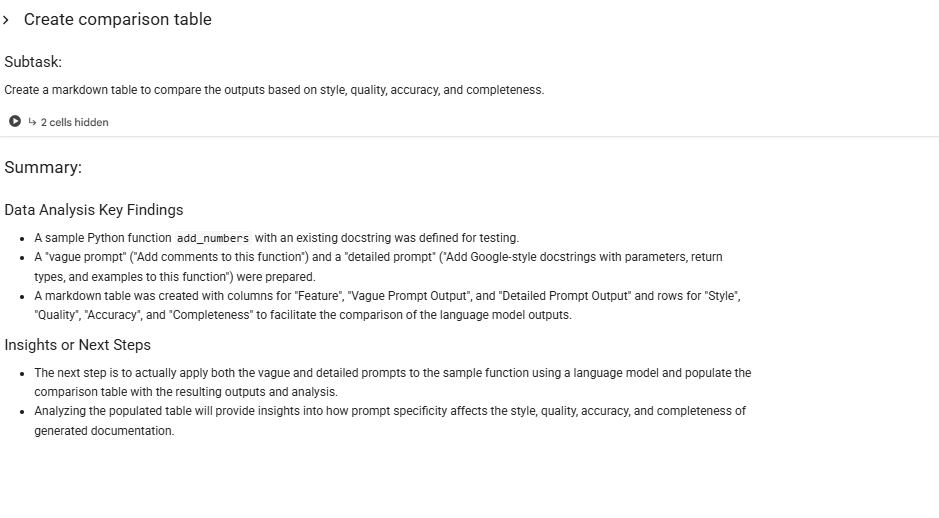
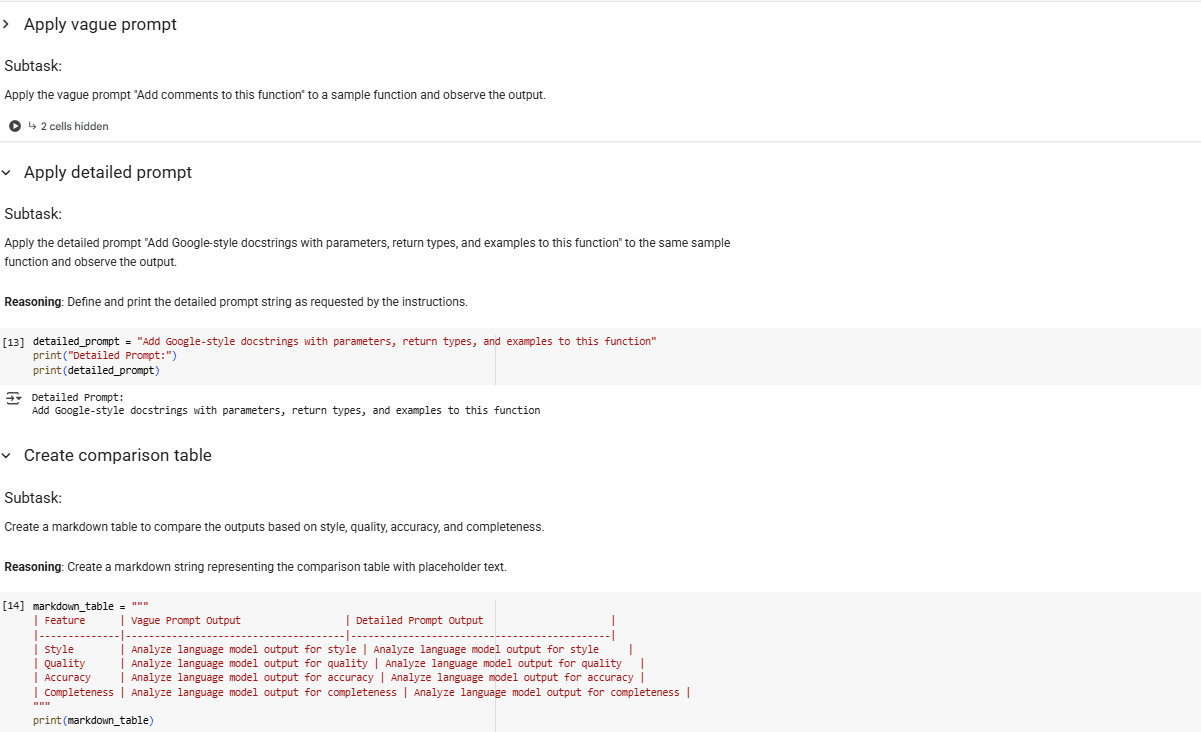
* **Task: Use AI to identify and correct inaccuracies in existing docstrings.**
* **Instructions:**
  + **Provide Python code with outdated or incorrect docstrings.**
  + **Instruct AI to rewrite each docstring to match the current code behavior.**
  + **Ensure corrections follow Google-style formatting.**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

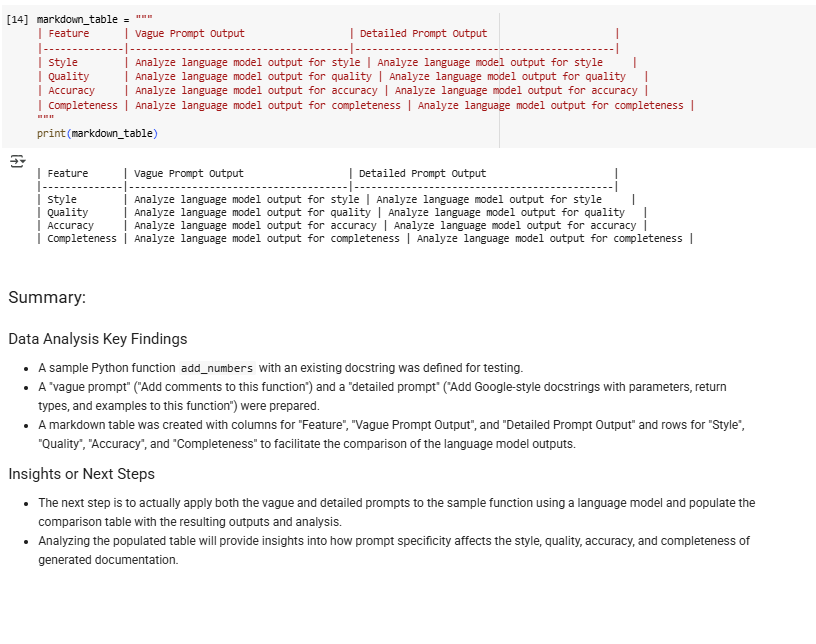
****

****

**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.
* Expected Output #6:

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

****