ASSIGNMENT:9.2

HTNO:2403A51284

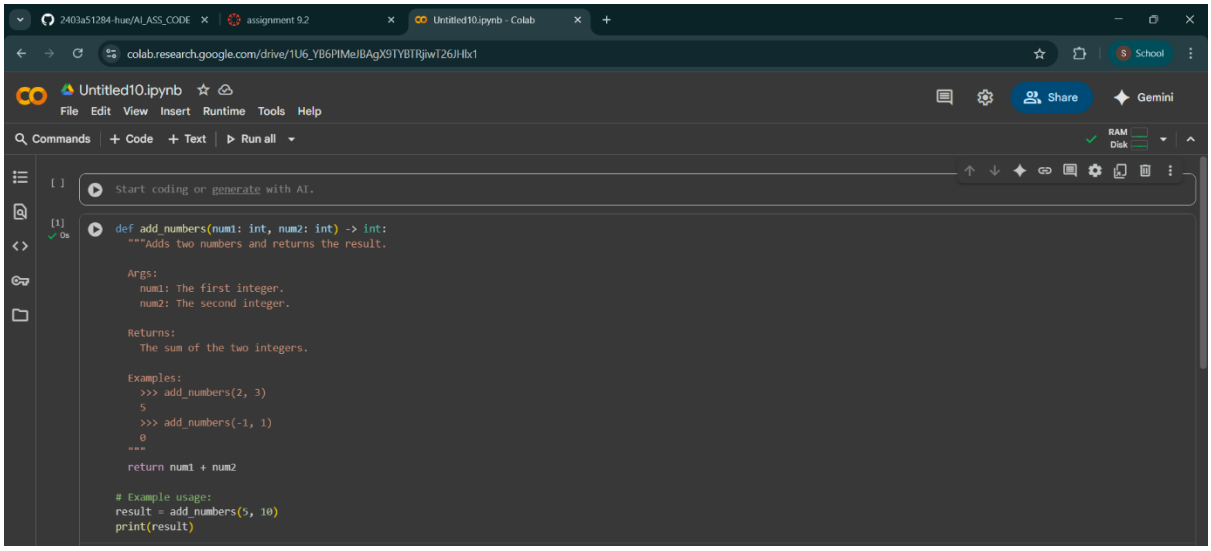
BATCH:12

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

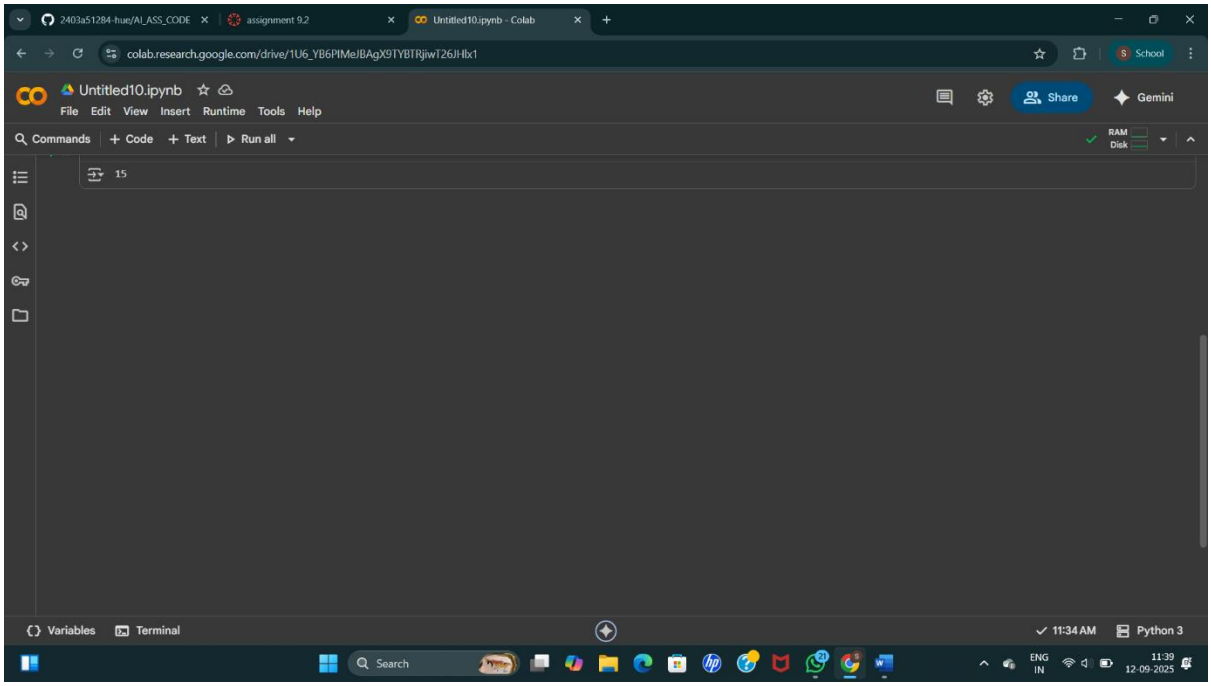
PROMPT:

- A brief function description
- Parameters with type hints
- Return values with type hints
- Example usage (without input-output examples)
  Ensure the docstrings are accurate and correctly formatted.

CODE:



OUTPUT:



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.

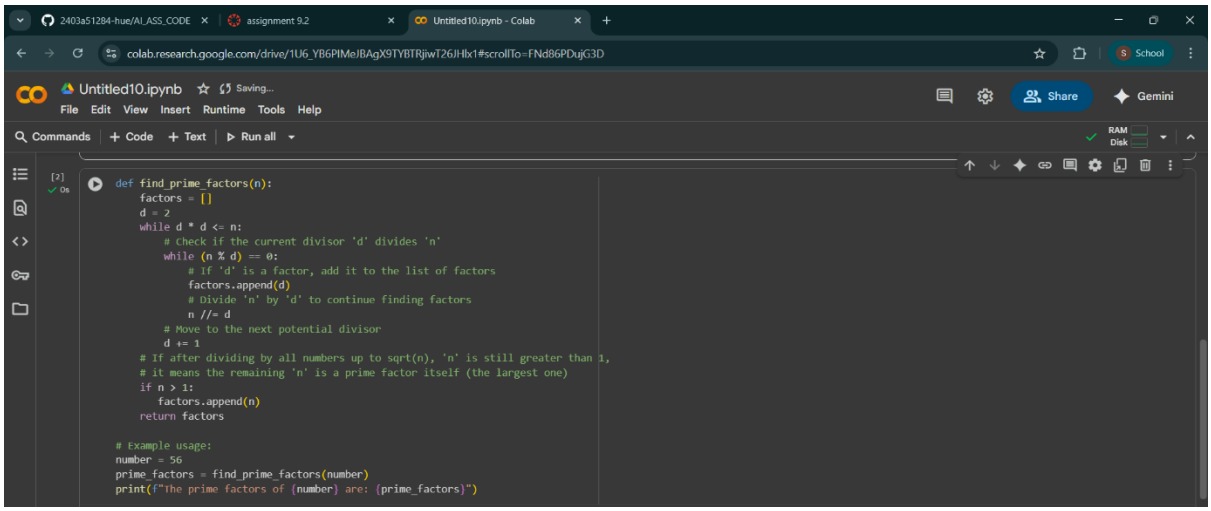
• Expected Output #2:

o Python code with concise, context-aware inline comments for complex logic blocks

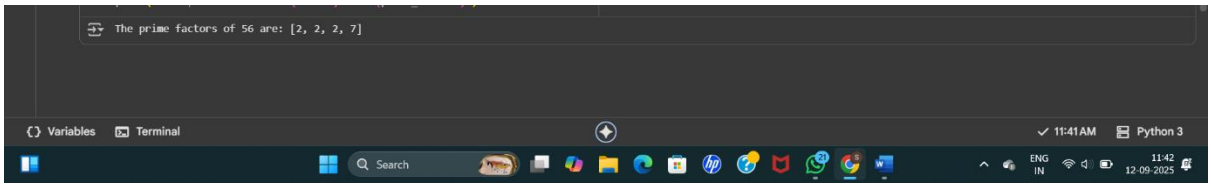
PROMPT:

Add meaningful inline comments to the following Python script, explaining only the complex or non-intuitive logic parts.
Skip comments for obvious syntax or straightforward code.
Focus on tricky sections to improve readability and maintainability

CODE:



OUTPUT:

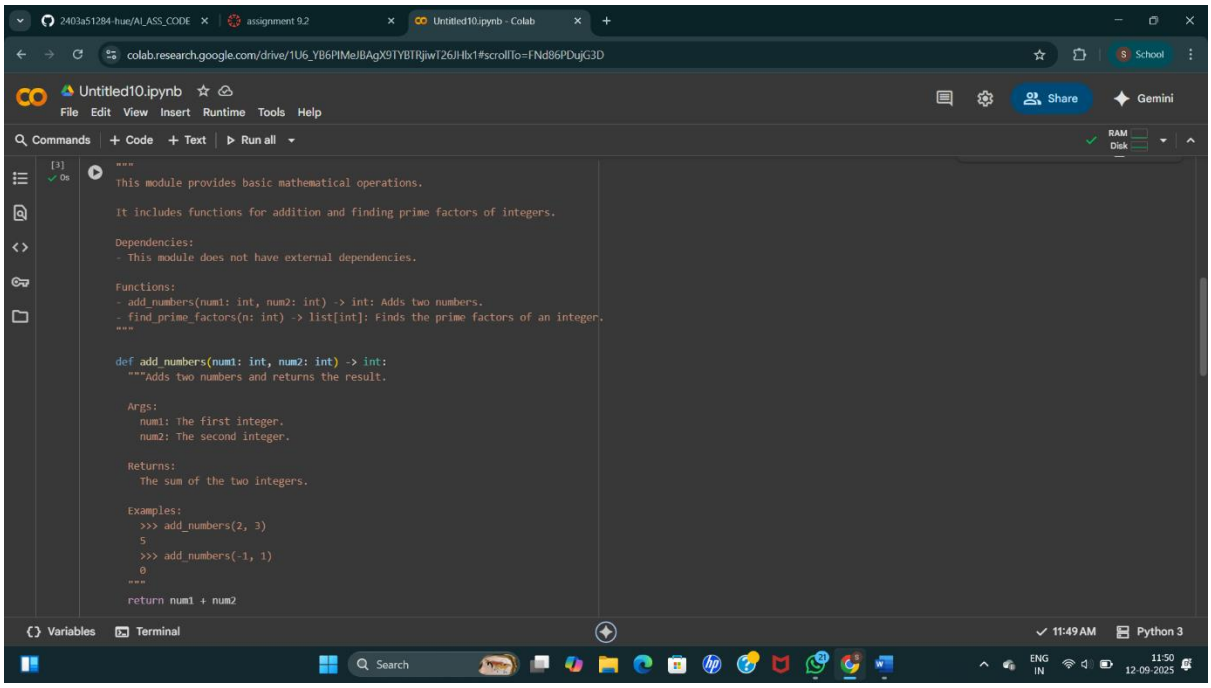


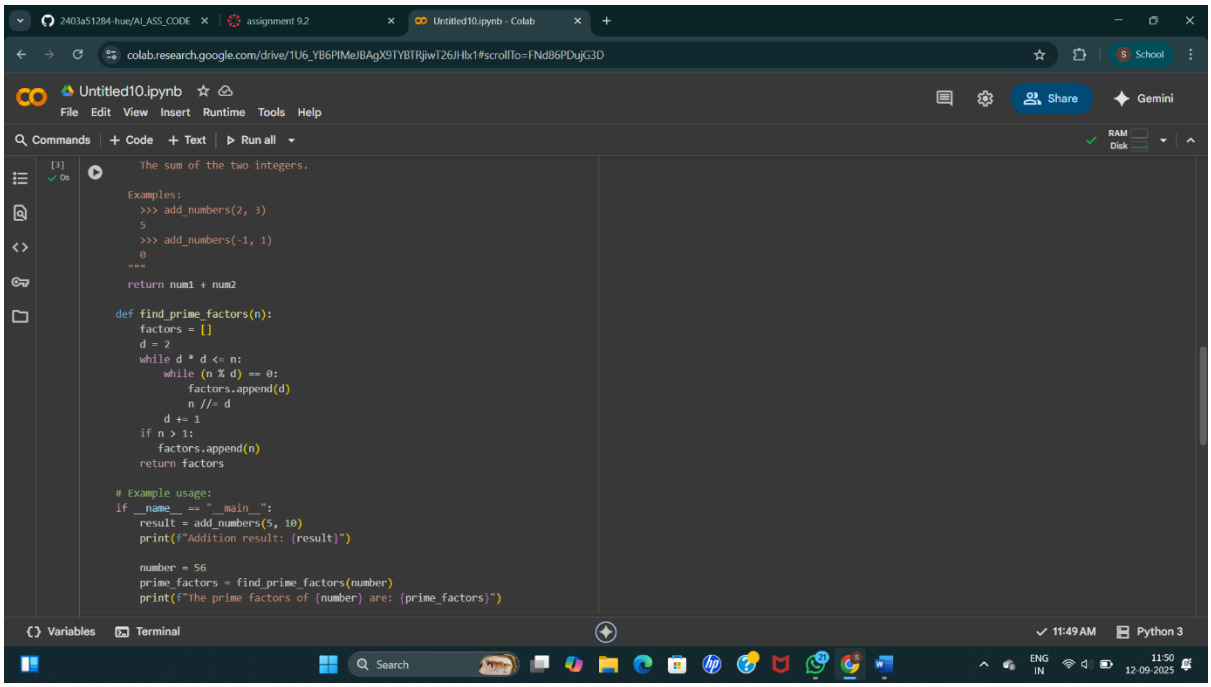
Task Description #3 (Documentation – Module-Level Documentation)
• Task: Use AI to create a module-level docstring summarizing the purpose, dependencies, and main functions/classes of a Python file. • Instructions:
o Supply the entire Python file to AI.
o Instruct AI to write a single multi-line docstring 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.
The docstring should summarize the file's purpose, dependencies, and main functions/classes.
Do not rewrite the code; just provide a clear, concise description of functionality and usage.

CODE:





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:

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.

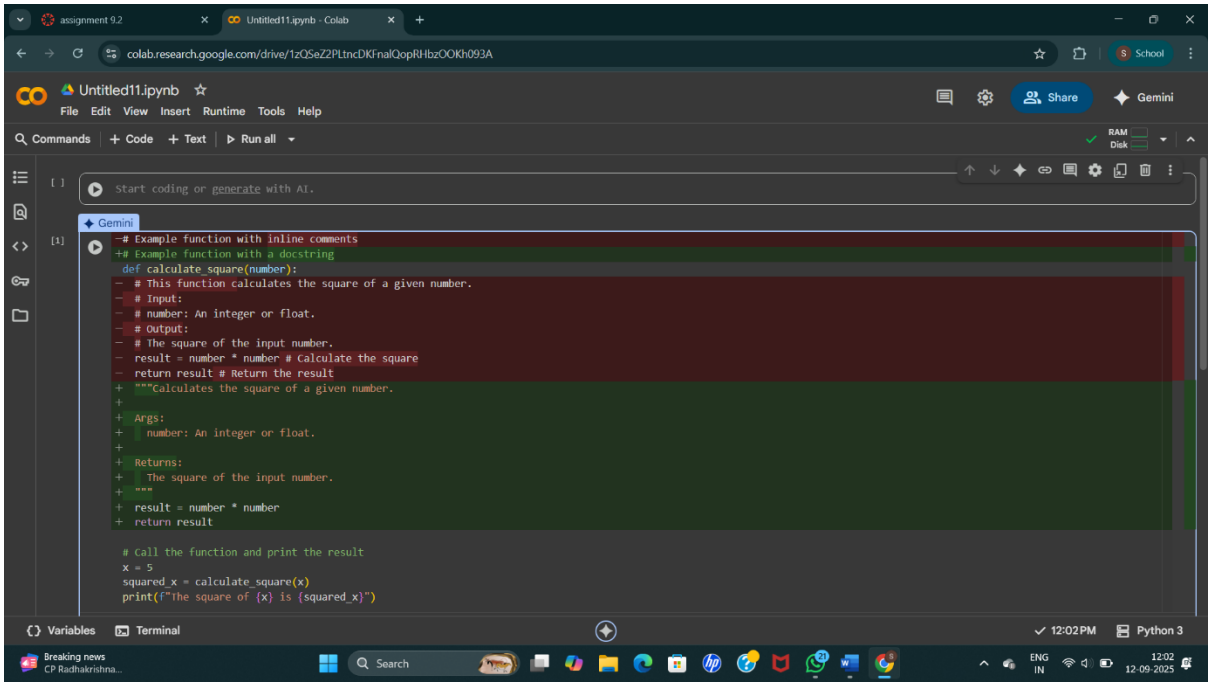
• Expected Output #4:

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

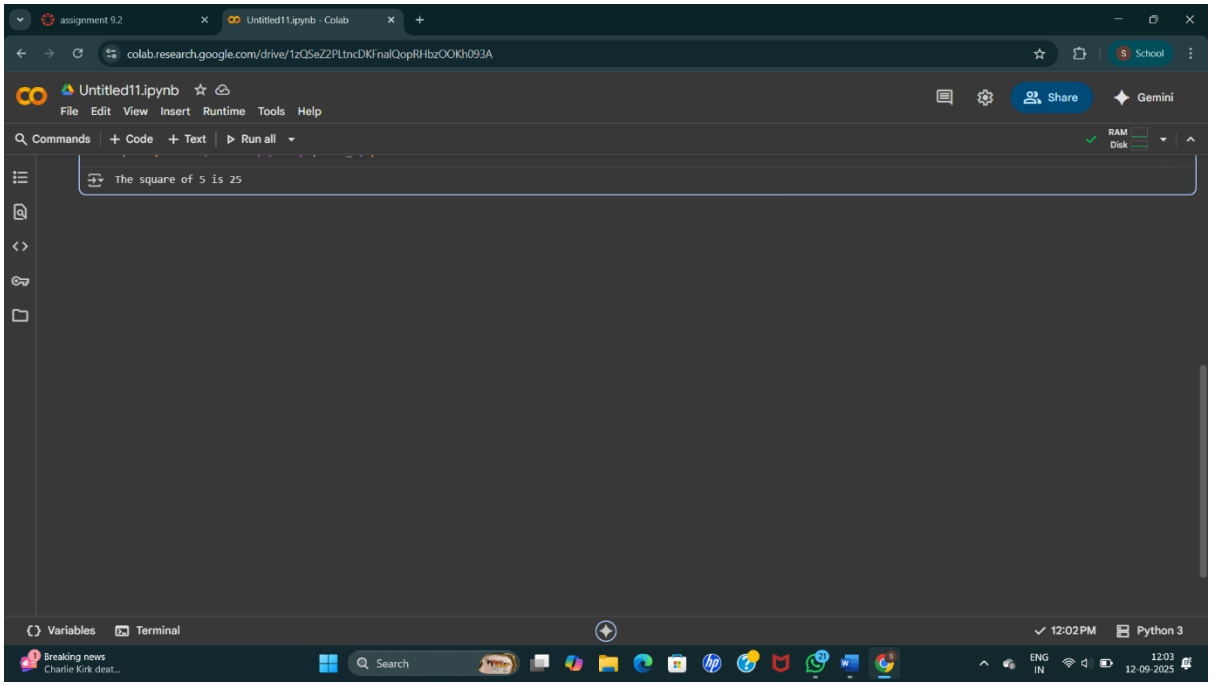
PROMPT:

Transform all existing inline comments in the following Python code into structured function docstrings using Google style.
Move relevant details from comments into the docstrings, ensuring the meaning is preserved and the documentation is clear and standardized.

CODE:



OUTPUT:



o Python file with updated, accurate, and standardized Task Description #5 (Documentation – Review and Correct
Docstrings)

• Task:5 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.

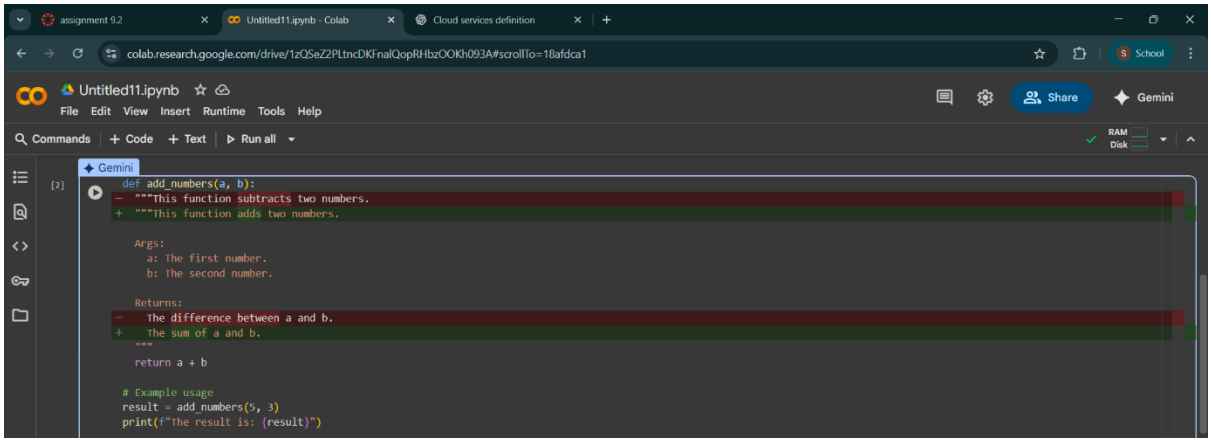
o Ensure corrections follow Google-style formatting.
• Expected Output #5:

docstrings

PROMPT:

Review the following Python code and identify any outdated or incorrect docstrings.
Rewrite each docstring to accurately reflect the current code behavior, using Google-style formatting for all corrections.

CODE:



OUTPUT:

Description #6 (Documentation – Prompt Comparison

Task (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

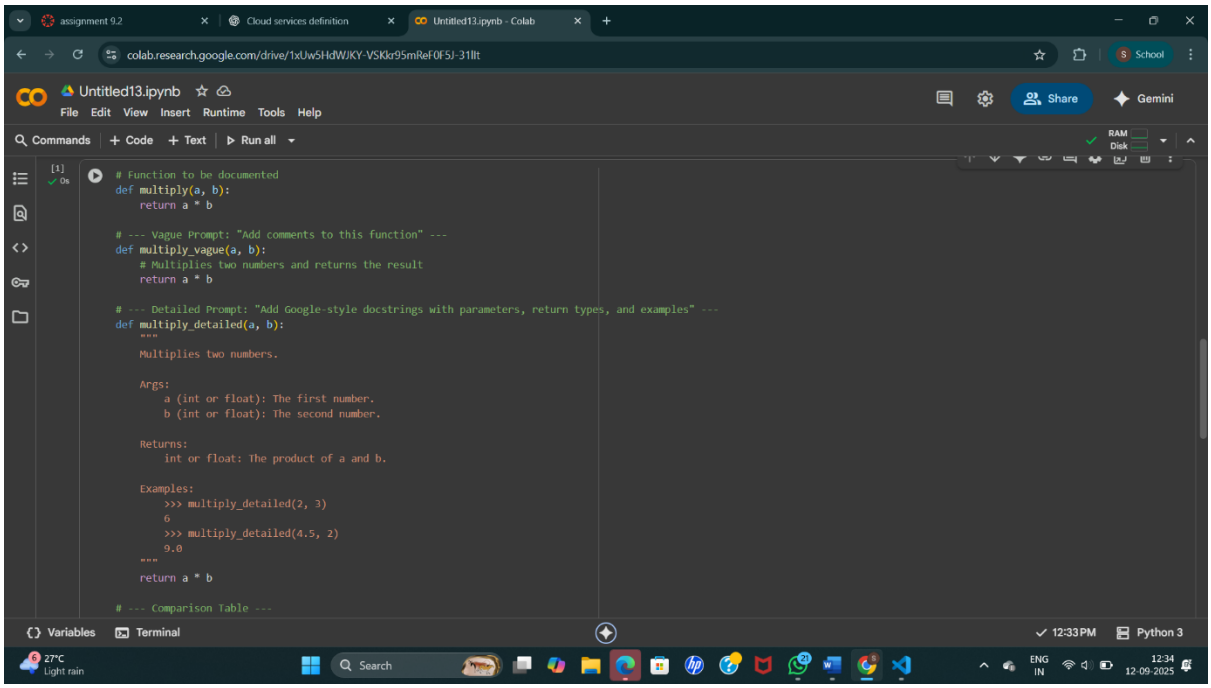
completeness.
• Expected Output #6:
o A comparison table showing the results from both
prompts with observations.

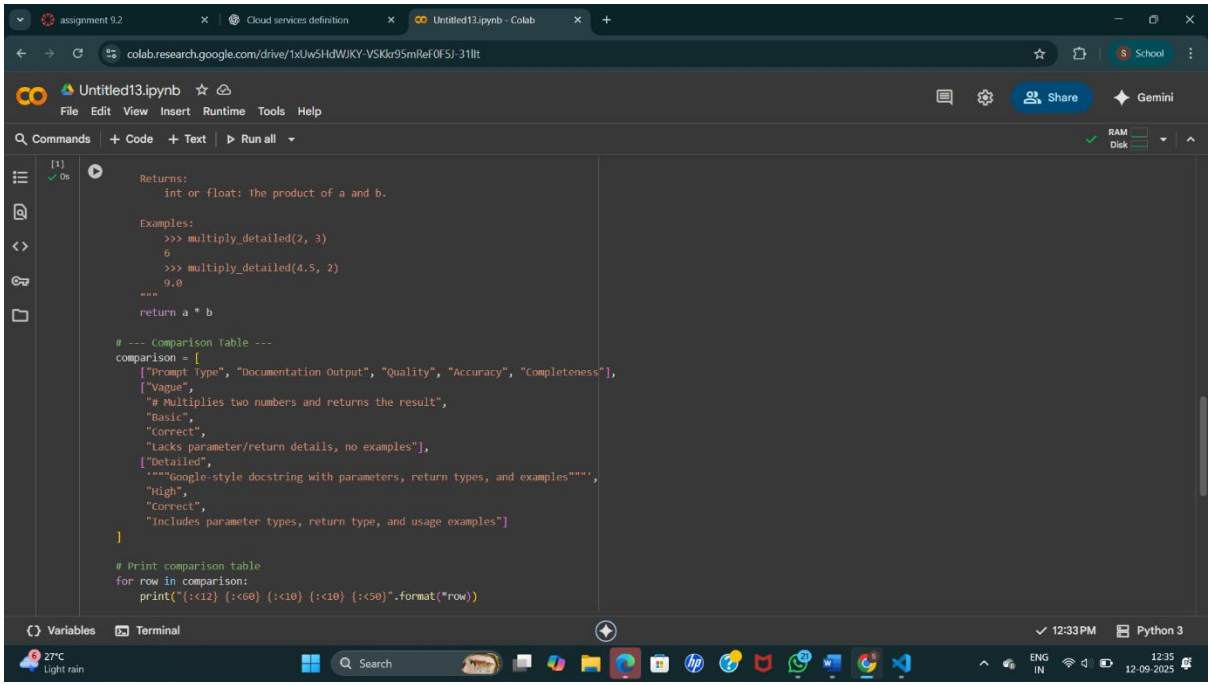
PROMPT:

Prompt 1 (Vague):
Add comments to this function.

Prompt 2 (Detailed):
Add Google-style docstrings to this function, including a brief description, parameter types, return type, and example usage.

CODE:




assignment 9.2
Cloud services definition
Untitled13.ipynb - Colab
colab.research.google.com/drive/1xUw5HdWJKY-VSKkr95mReF0F5J-31llt
Untitled13.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM
Disk
[1]
0s
Returns:
int or float: The product of a and b.
Examples:
>>> multiply_detailed(2, 3)
6
>>> multiply_detailed(4.5, 2)
9.0
"""
return a * b
# --- Comparison Table ---
comparison = [
["Prompt Type", "Documentation Output", "Quality", "Accuracy", "Completeness"],
["Vague",
"# Multiplies two numbers and returns the result",
"Basic",
"Correct",
"Lacks parameter/return details, no examples"],
["Detailed",
'"""Google-style docstring with parameters, return types, and examples"""',
"High",
"Correct",
"Includes parameter types, return type, and usage examples"]
]
# Print comparison table
for row in comparison:
print("{:<12} {:<60} {:<10} {:<10} {:<50}".format(*row))
Variables
Terminal
12:33PM
Python 3
27°C
Light rain
Search
ENG
IN
12:35
12-09-2025

OUTPUT:


Prompt Type Documentation Output Quality Accuracy Completeness
Vague # Multiplies two numbers and returns the result Basic Correct Lacks parameter/return details, no examples
Detailed """Google-style docstring with parameters, return types, and examples""" High Correct Includes parameter types, return type, and usage examples