HINO:2403A51286  
 Assignment:9.5

**Task#1 (Automatic Code Commenting)**

**Scenario:** You have been given a Python function without comments.

def calculate\_discount(price, discount\_rate):

return price - (price \* discount\_rate / 100)

• Use an AI tool (or manually simulate it) to generate line-by-line

comments for the function.

• Modify the function so that it includes a docstring in Google-style

or NumPy-style format.

• Compare the auto-generated comments with your manually

written version **Prompt:**

You are given the following Python function without comments:

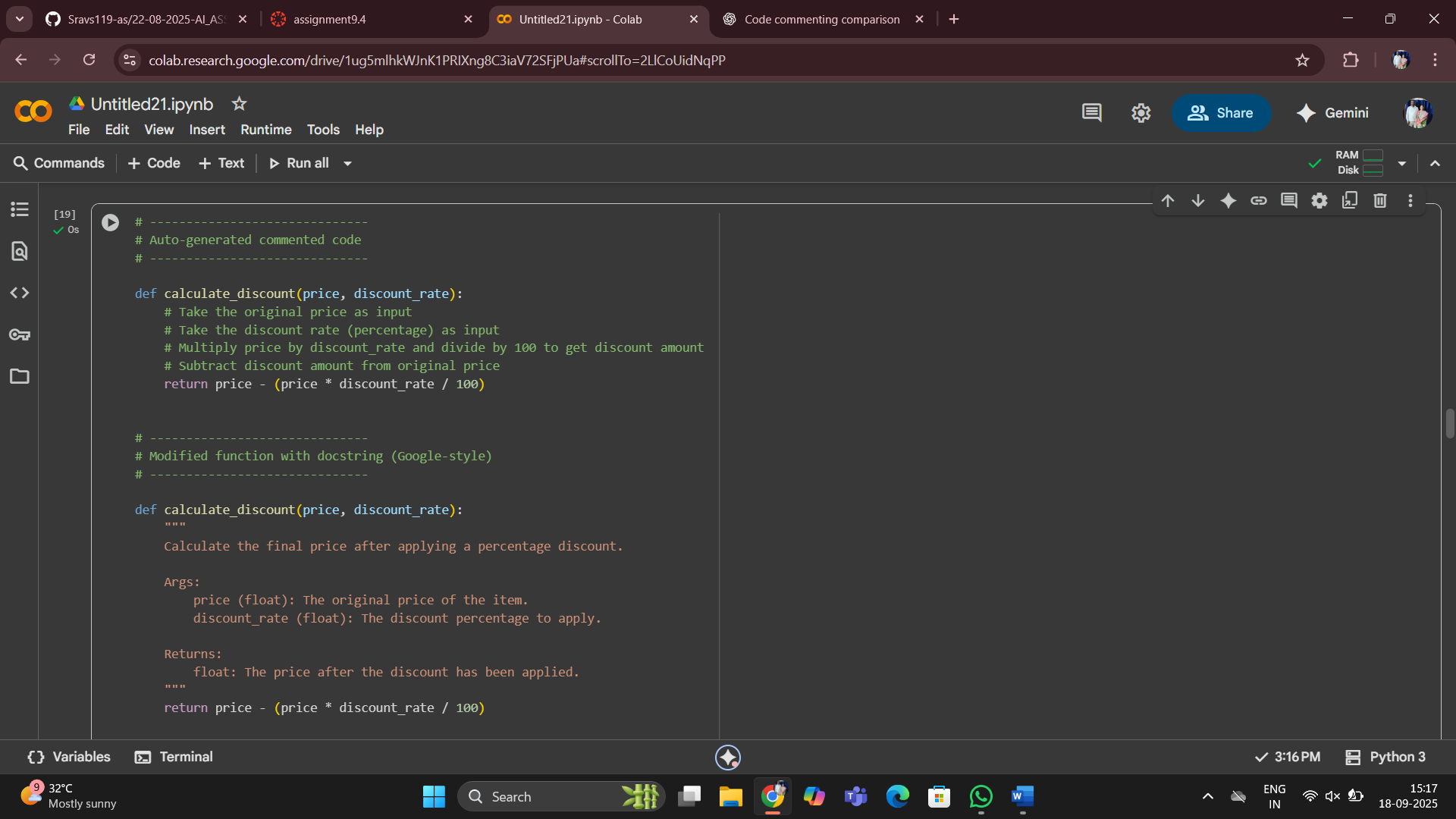
def calculate\_discount(price, discount\_rate):

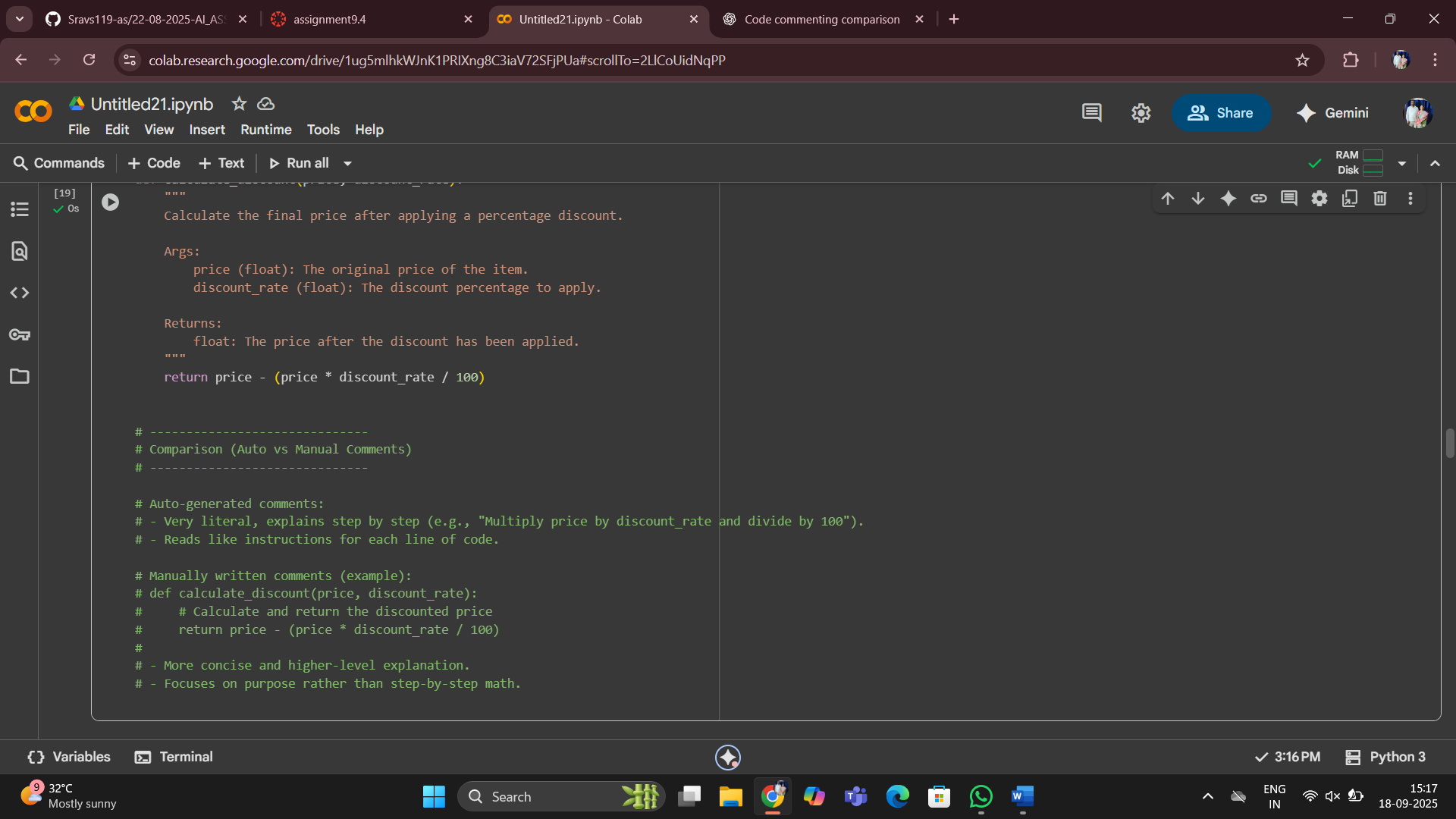
return price - (price \* discount\_rate / 100)

**Your tasks are:**

1. Generate **line-by-line comments** for this function.
2. Rewrite the function to include a **docstring** using **Google-style** or **NumPy-style** format.
3. Provide a **comparison** between the AI-generated comments and manually written comments (highlight similarities and differences).

**Code:**

****



**Task#2(API Documentation Generator)  
Scenario:** A team is building a Library Management System with  
multiple functions.  
def add\_book(title, author, year):  
# code to add book  
pass  
def issue\_book(book\_id, user\_id):  
# code to issue book  
Pass  
• Write a Python script that uses docstrings for each function (with  
input, output, and description).  
• Use a documentation generator tool (like pdoc, Sphinx, or  
MkDocs) to automatically create HTML documentation.  
• Submit both the code and the generated documentation as outpu  
**Prompt:**

You are given two Python functions for a Library Management System:

def add\_book(title, author, year):

# code to add book

pass

def issue\_book(book\_id, user\_id):

# code to issue book

pass

Tasks:

1. Add docstrings to each function in Google-style or NumPy-style, including input parameters, return values, and description.
2. Write a Python script containing these functions.
3. Show how to use a documentation generator tool (e.g., pdoc, Sphinx, or MkDocs) to automatically generate HTML documentation from the docstrings.
4. Provide both the updated Python code and an example of the generated documentation output (in text form or HTML snippet).

**Code:**

****

**Task#3 (AI-Assisted Code Summarization)  
Scenario**: You are reviewing a colleague’s codebase containing long  
functions.

def process\_sensor\_data(data):  
cleaned = [x for x in data if x is not None]  
avg = sum(cleaned)/len(cleaned)  
anomalies = [x for x in cleaned if abs(x - avg) > 10]  
return {"average": avg, "anomalies": anomalies}  
• Generate a summary comment explaining the purpose of the  
function in 2–3 lines.  
• Create a flow-style comment (step-by-step explanation).  
• Write a short paragraph of documentation describing possible use  
cases of this function in real-world scenarios.

**Prompt:**

You are given the following Python function:

def process\_sensor\_data(data):

cleaned = [x for x in data if x is not None]

avg = sum(cleaned)/len(cleaned)

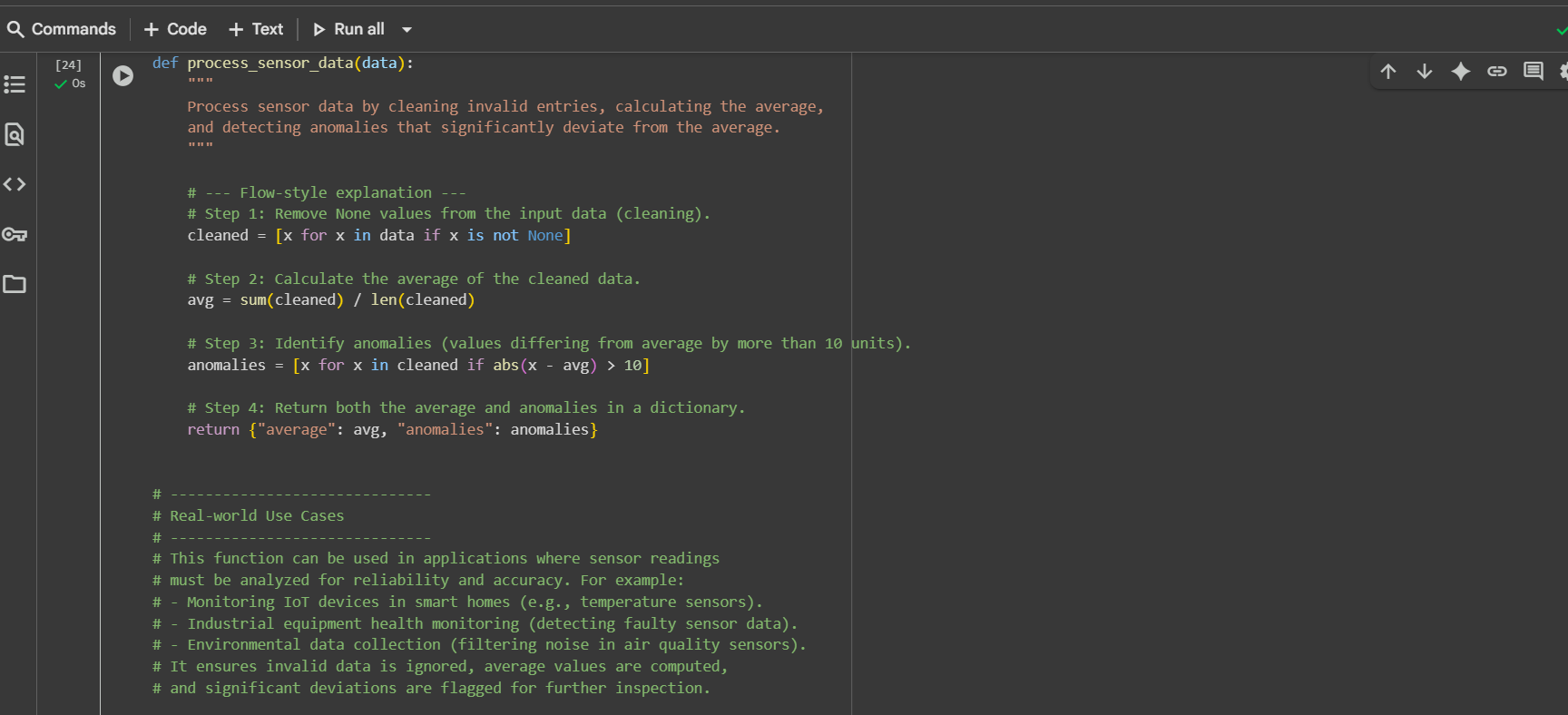
anomalies = [x for x in cleaned if abs(x - avg) > 10]

return {"average": avg, "anomalies": anomalies}

Tasks:

1. Generate a summary comment (2–3 lines) explaining the overall purpose of the function.
2. Write a flow-style comment with a step-by-step explanation of what the function does.
3. Provide a short paragraph of documentation describing possible real-world use cases of this function.
4. Show the final updated code including all comments/docstring.

**Code:**

****

**Task#4(Real-Time Project Documentation)  
Scenario:** You are part of a project team that develops a Chatbot  
Application. The team needs documentation for maintainability.  
• Write a README.md file for the chatbot project (include project  
description, installation steps, usage, and example).  
• Add inline comments in the chatbot’s main Python script (focus  
on explaining logic, not trivial code).  
• Use an AI-assisted tool (or simulate it) to generate a usage guide  
in plain English from your code comments.  
• Reflect: How does automated documentation help in real-time  
projects compared to manual documentation?

**Prompt:**

You are part of a project team developing a Chatbot Application.

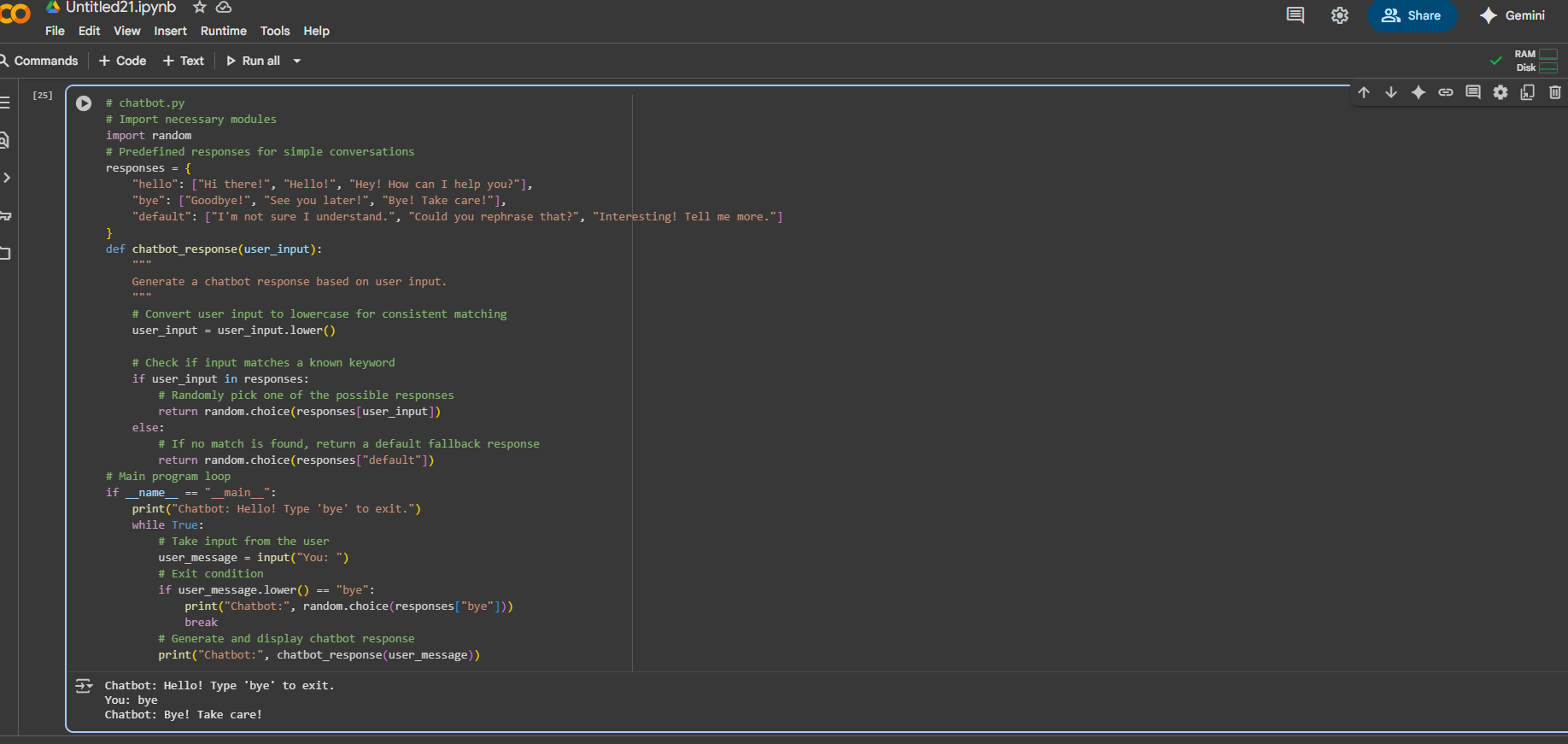
Your tasks are:

1. Write a README.md file for the chatbot project that includes:
   * Project description
   * Installation steps
   * Usage instructions
   * Example input/output conversation
2. Add inline comments to the chatbot’s main Python script. Focus on explaining the logic and flow, not trivial code.
3. Use an AI-assisted tool (or simulate it) to generate a plain-English usage guide from the code comments.
4. Write a reflection comparing automated documentation with manual documentation in real-time projects.

Finally, provide:

* The complete README.md file
* The Python script with inline comments
* The generated usage guide
* The reflection section

**Code:**

****