AI ASSISTED CODING LAB

ASSIGNMENT-9.1/9.4

ENROLLMENT NO:2503A51L20

BATCH NO: 19

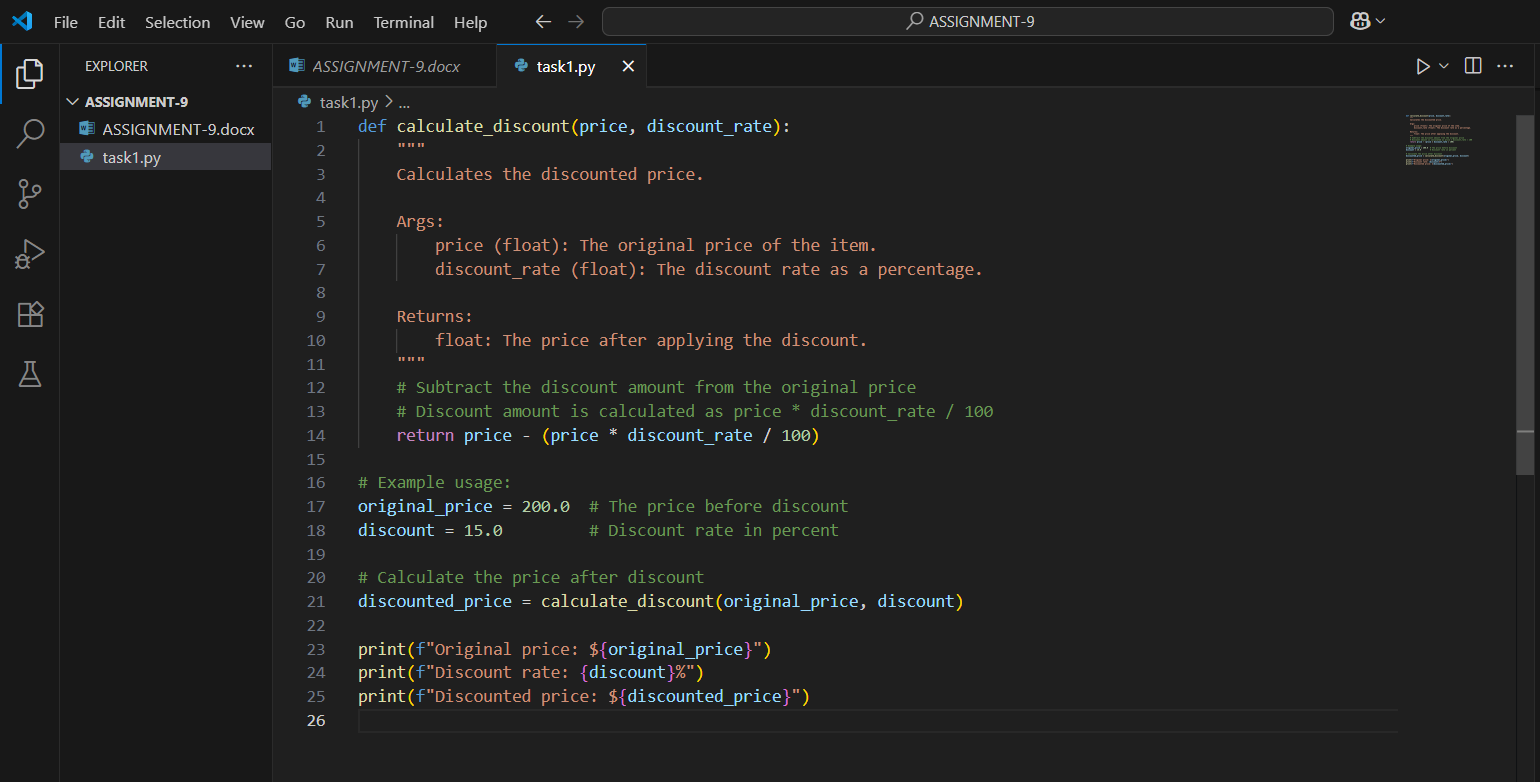
NAME: Guangsinlung Phaomei

TASK DESCRIPTION 1:

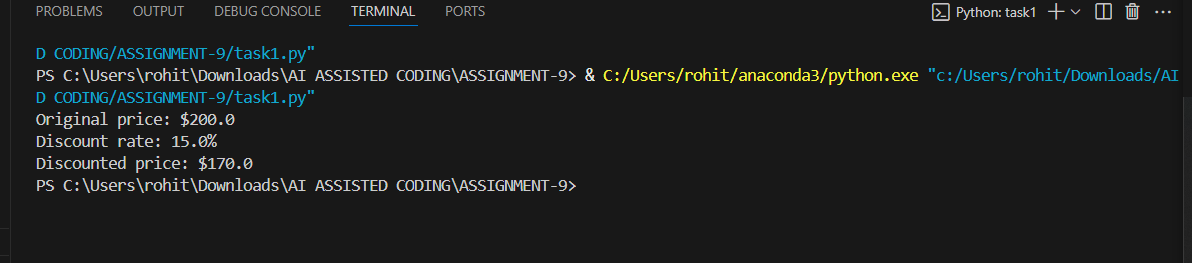
Scenario: You have been given a Python function without comments.  
def calculate discount(price, discount\_rate):  
return price - (price \* discount\_rate / 100)  
• Use an AI tool (or manually simulate it) to generate line-by-line  
comments for the function.  
• Modify the function so that it includes a docstring in Google-style  
or NumPy-style format.  
• Compare the auto-generated comments with your manually  
written version

PROMPT 1: Write a Python function that takes the original price and a discount rate, then calculates and returns the final price after applying the discount. Also include an example usage.

CODE GENERATED:



OUTPUT:



OBSERVATION: In this assignment, I learned how to analyse a simple Python function and enhance its readability by adding comments and a proper docstring. Using an AI tool (or simulating one), I generated line-by-line comments and then compared them with my own manually written comments. I observed that while AI-generated comments are often concise and technical, my own comments could be more detailed and contextual, making the code easier to understand for beginners. Adding a Google-style or NumPy-style docstring also helped in clearly defining the purpose, parameters, and return value of the function. Overall, this task improved my understanding of writing clean, well-documented, and user-friendly code.

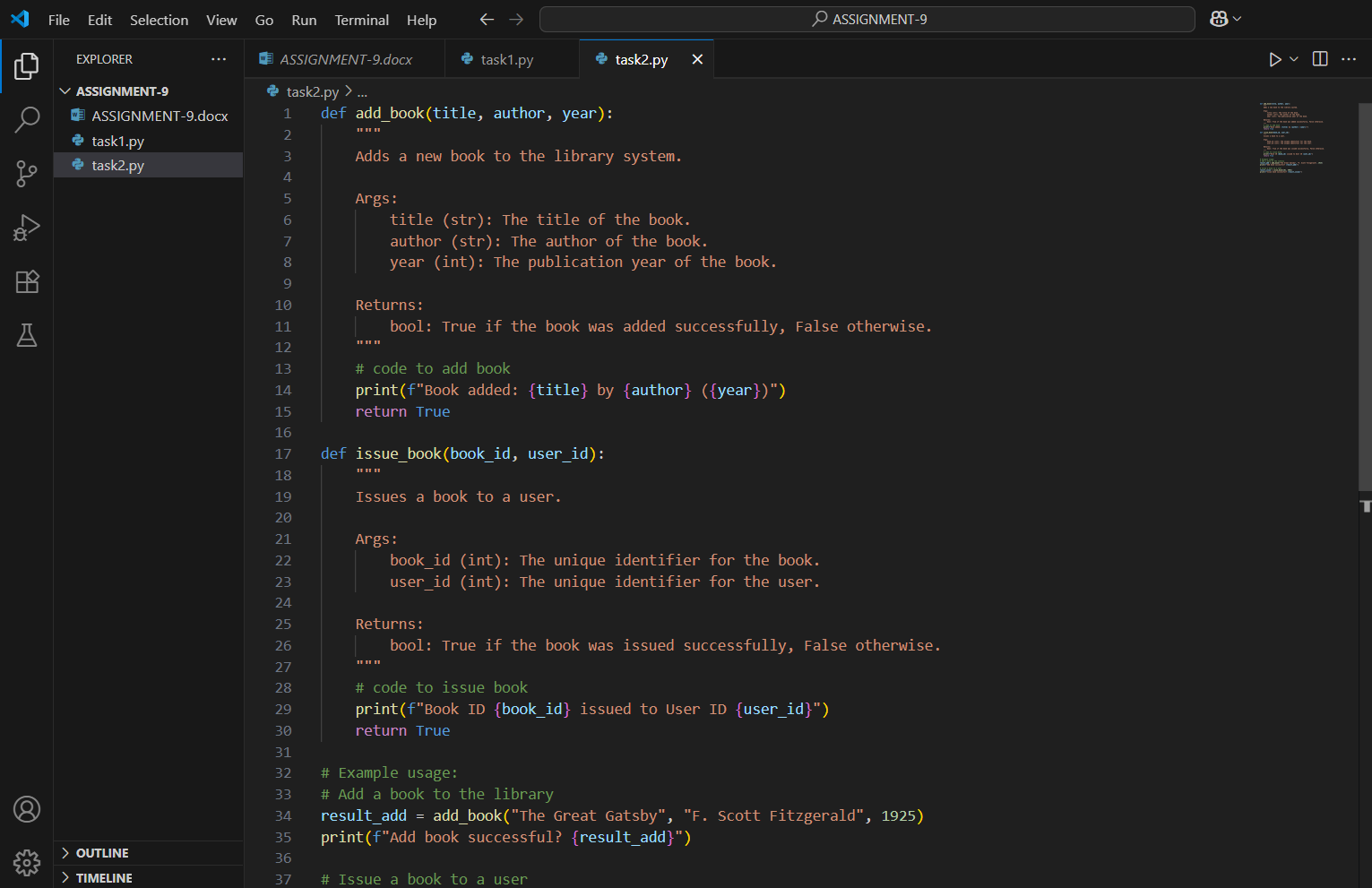
TASK DESCRIPTION 2:

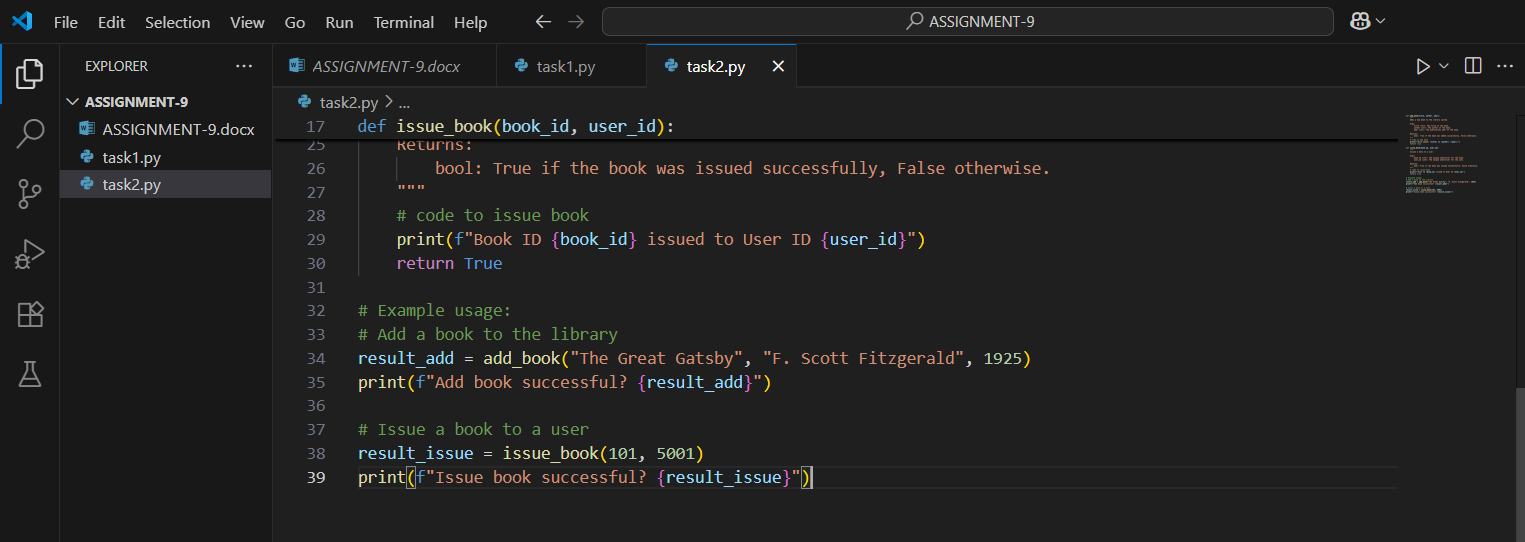
Scenario: A team is building a Library Management System with  
multiple functions.  
def add\_book(title, author, year):  
# code to add book  
pass  
def issue\_book(book\_id, user\_id):  
# code to issue book  
Pass  
• Write a Python script that uses docstrings for each function (with  
input, output, 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 output.

PROMPT 1: Write a Python program for a Library Management System.

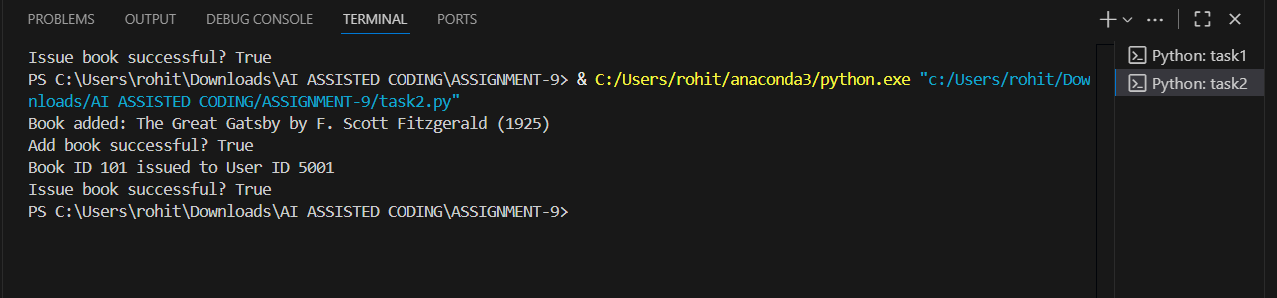
* Make two functions:
  1. add\_book(title, author, year) → to add a book.
  2. issue\_book(book\_id, user\_id) → to issue a book.
* Write clear **docstrings** (description, inputs, outputs) for both functions.
* Use a tool like **pdoc, Sphinx, or MkDocs** to create **HTML documentation** from your docstrings.
* Submit the **Python code** and the **HTML documentation** as output.

CODE GENERATED:





OUTPUT:



OBSERVATION : In this assignment, I created a simple Python program for a Library Management System with two main functions: add\_book() and issue\_book(). Each function was documented using clear docstrings that explained the purpose, input parameters, and expected output.

I observed that writing proper docstrings not only makes the code more understandable but also allows documentation generator tools (like **pdoc**, **Sphinx**, or **MkDocs**) to automatically produce structured HTML documentation. This helps in maintaining large projects where multiple developers are involved, as the generated documentation provides a quick reference without needing to read the full code.

Through this task, I understood the importance of consistent documentation practices in software development and how automated tools can save time by converting docstrings into professional, ready-to-use documentation.

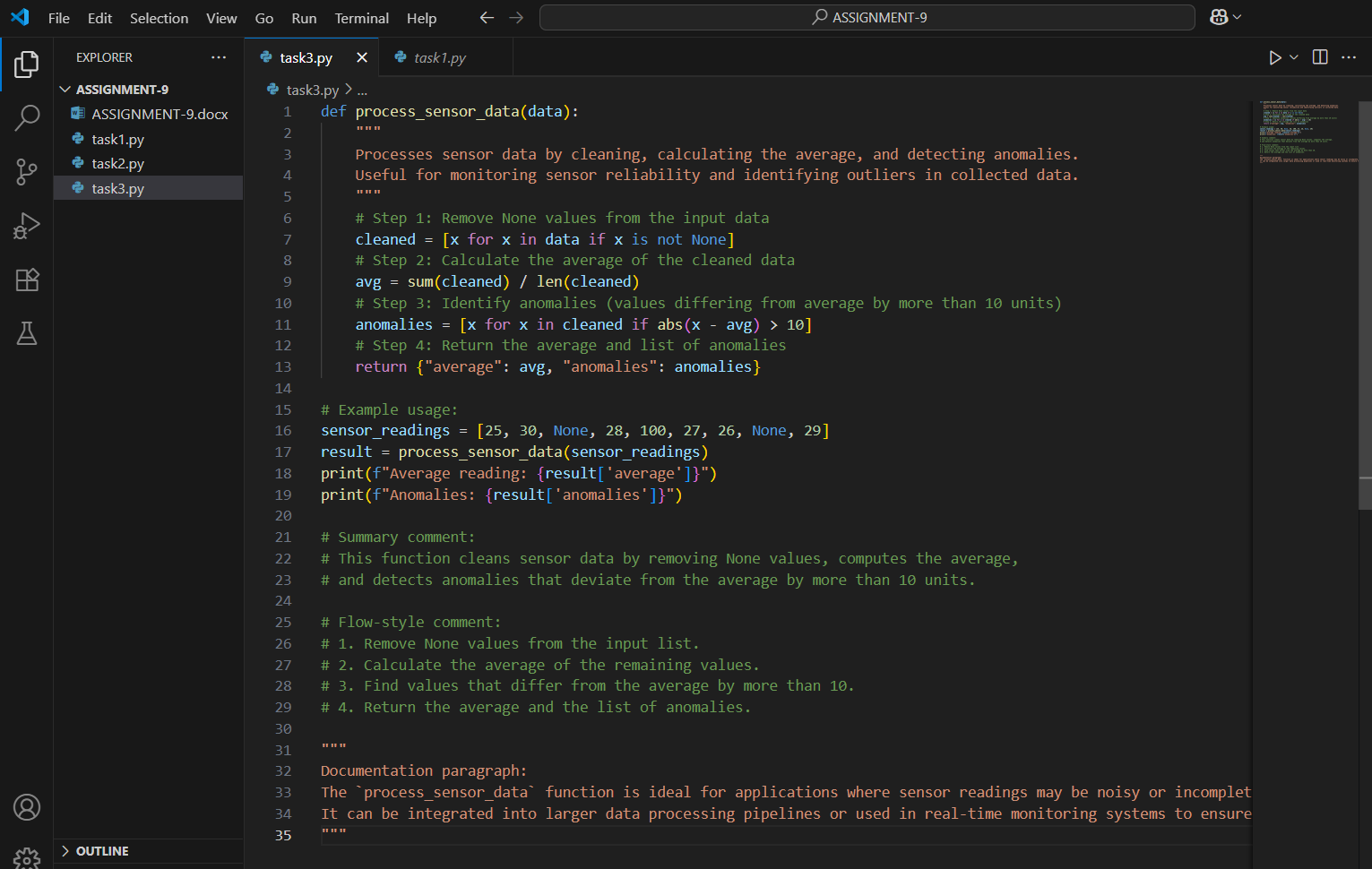
TASK DESCRIPTION 3: Scenario: You are reviewing a colleague’s codebase containing long  
functions.

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

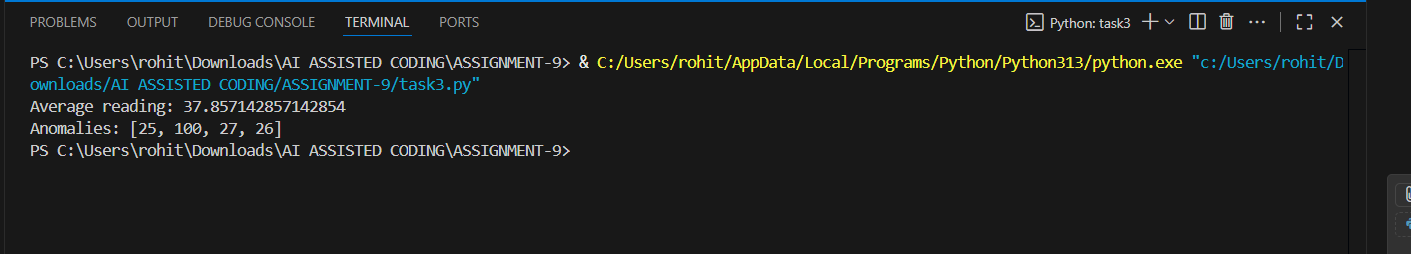
PROMPT 1: Review the function process\_sensor\_data(data).

* Write a 2–3 line summary explaining what the function does.
* Add step-by-step (flow-style) comments inside the function.
* Write a short paragraph describing real-world uses, such as sensor monitoring or anomaly detection.

CODE GENERATED:



OUTPUT:



OBSERVATION:

In this assignment, I reviewed the function process\_sensor\_data(data) and created a concise summary explaining its purpose: to clean a dataset, calculate the average, and detect anomalies. I added flow-style comments to explain each step of the function, making it easier to understand how the data is processed. Additionally, I wrote a short paragraph describing real-world use cases, such as monitoring sensor readings, detecting abnormal values in IoT systems, or analyzing data for predictive maintenance.

This exercise highlighted the importance of clear documentation and comments, especially in long functions, as it improves code readability, maintainability, and helps other developers quickly understand the logic and potential applications of the function.

TASK DESCRIPTION 4: 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?

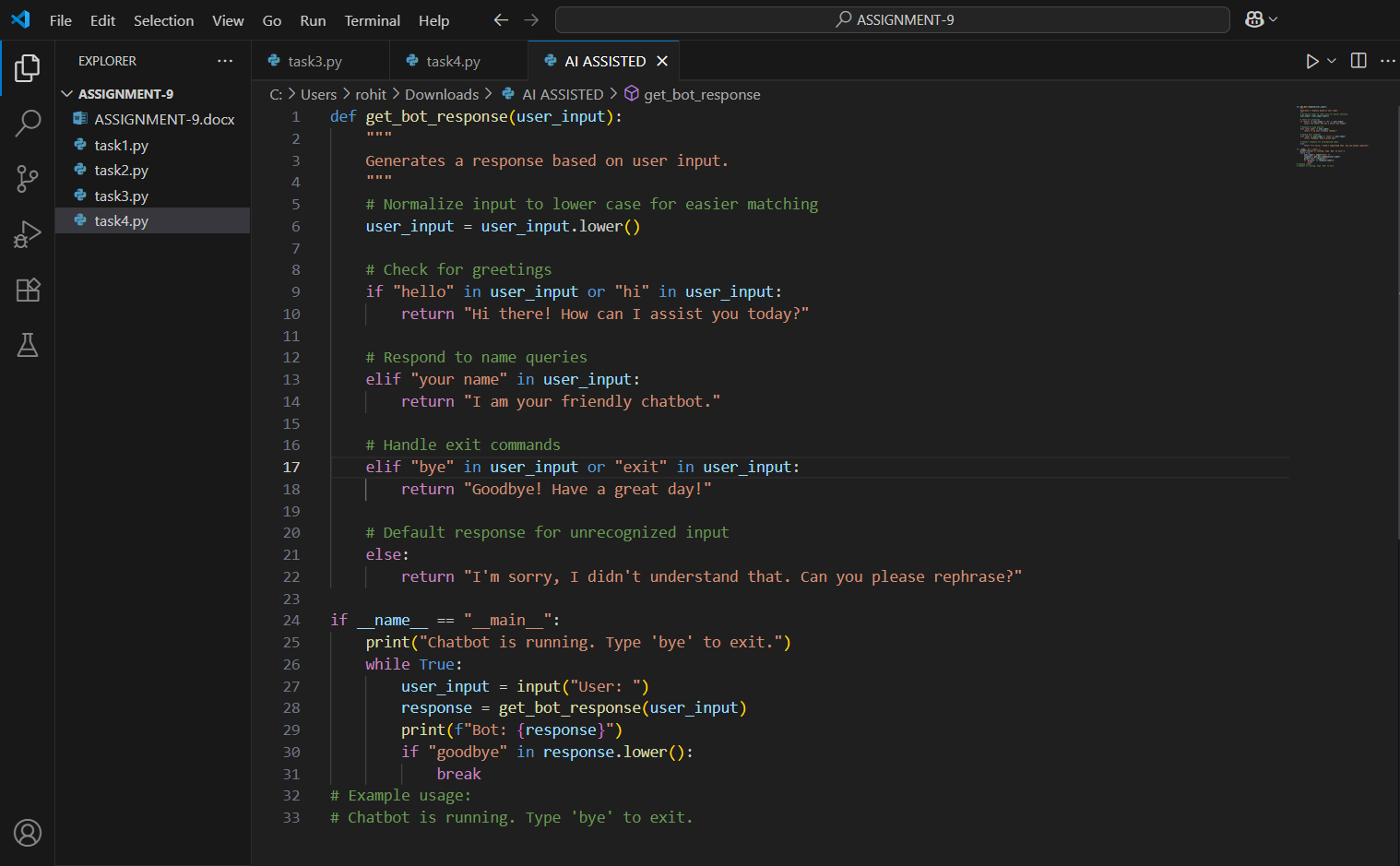
PROMPT 1:

Create a **README.md** file for the project, including: project description, installation steps, usage instructions, and an example.

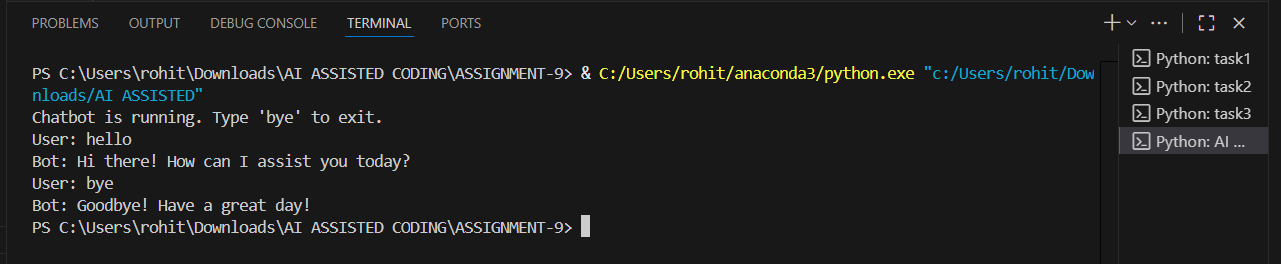
Add **inline comments** in the main Python script, explaining the logic of the code (not trivial lines).

Use an **AI-assisted tool** (or simulate it) to generate a plain English **usage guide** from your code comments.

Write a short **reflection** on how automated documentation is helpful in real-time projects compared to manual documentation.

CODE GENERATED:

OUTPUT:



OBSERVATION: In this assignment, I created a README.md file for the Chatbot project that included a project description, installation steps, usage instructions, and an example. I also added inline comments in the main Python script to explain the logic of the code, making it easier for others to understand and maintain. Using an AI-assisted tool, I generated a plain English usage guide from the code comments, which provided a clear, user-friendly reference.

Through this task, I observed that automated documentation significantly improves efficiency in real-time projects by quickly producing consistent and readable guides, reducing errors, and saving time compared to manually writing all documentation.