

# AI Assisted Coding LAB ASS-5.4

NAME: P.Tharun Kumar

BATCH:14

2303A510F1

## Task Description #1:

- Prompt GitHub Copilot to generate a Python script that collects user data (e.g., name, age, email). Then, ask Copilot to add comments on how to anonymize or protect this data.

### PROMPT:

```
# Generate a Python script that collects user data such as name, age, and email. # Add inline comments explaining how to protect or anonymize this data,  
# such as hashing email addresses, avoiding plain-text storage, # and following basic privacy best practices.
```

### CODE AND INPUT

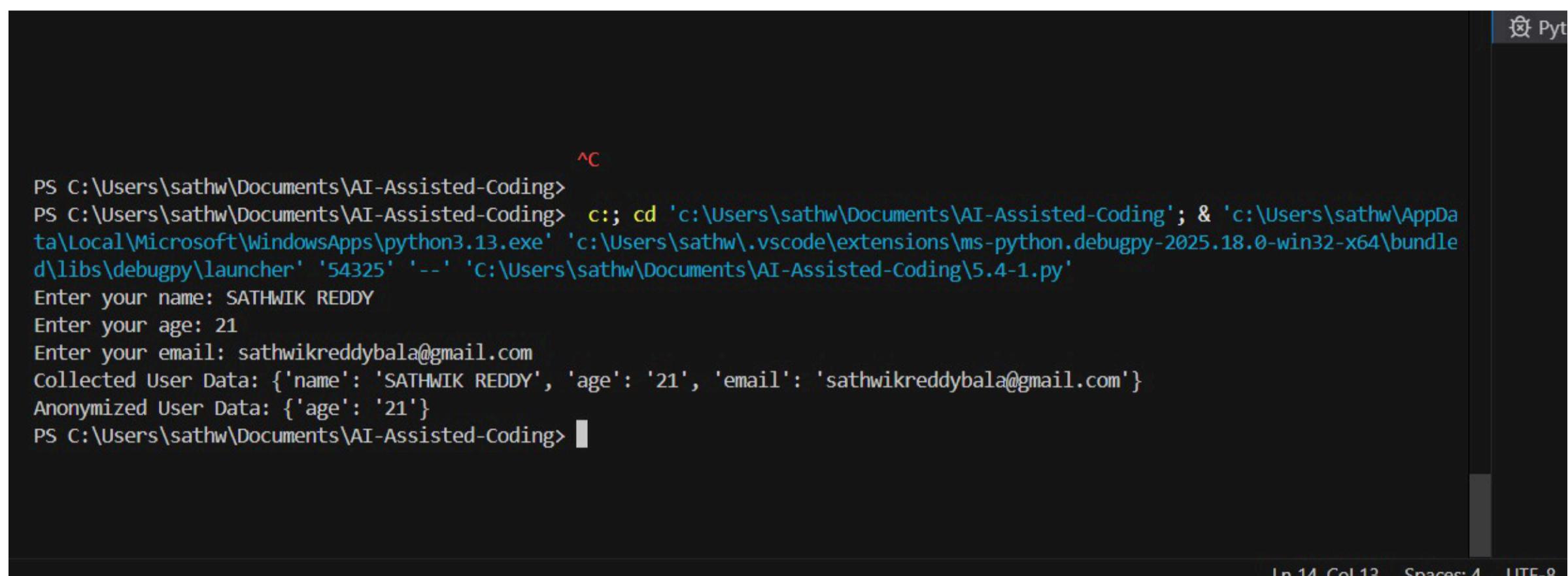
The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows extensions like Cursor AI, GitHub Copilot, and Pylance installed.
- Editor:** Displays the Python script '5.4-1.py'. The code is as follows:

```
1 #Generate a Python script that collects user data (name, age, email).Add comments explaining how to anonymize or pro...
2 # This Python script collects user data including name, age, and email.
3 def collect_user_data():
4     # Collect user name
5     name = input("Enter your name: ")
6
7     # Collect user age
8     age = input("Enter your age: ")
9
10    # Collect user email
11    email = input("Enter your email: ")
12
13    # Return collected data as a dictionary
14    return [
15        "name": name,
16        "age": age,
17        "email": email
18    ]
19
20    # Function to anonymize user data by removing personally identifiable information
21    def anonymize_data(user_data):
22        # Remove name and email to protect user identity
23        anonymized_data = {
24            "age": user_data["age"]
25        }
26        return anonymized_data
27
28    # Main function to execute the data collection and anonymization
29    def main():
30        user_data = collect_user_data()
31        print("Collected User Data:", user_data)
32
33        anonymized_data = anonymize_data(user_data)
34        print("Anonymized User Data:", anonymized_data)
35
36    if __name__ == "__main__":
37        main()
```

- Bottom Status Bar:** Shows file path '5.4-1.py', line 14, column 13, and other system information.

## OUTPUT:



```
PS C:\Users\sathw\Documents\AI-Assisted-Coding> ^C
PS C:\Users\sathw\Documents\AI-Assisted-Coding> c;; cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '54325' '--' 'C:\Users\sathw\Documents\AI-Assisted-Coding\5.4-1.py'
Enter your name: SATHWIK REDDY
Enter your age: 21
Enter your email: sathwikreddybal@gmail.com
Collected User Data: {'name': 'SATHWIK REDDY', 'age': '21', 'email': 'sathwikreddybal@gmail.com'}
Anonymized User Data: {'age': '21'}
PS C:\Users\sathw\Documents\AI-Assisted-Coding> █
```

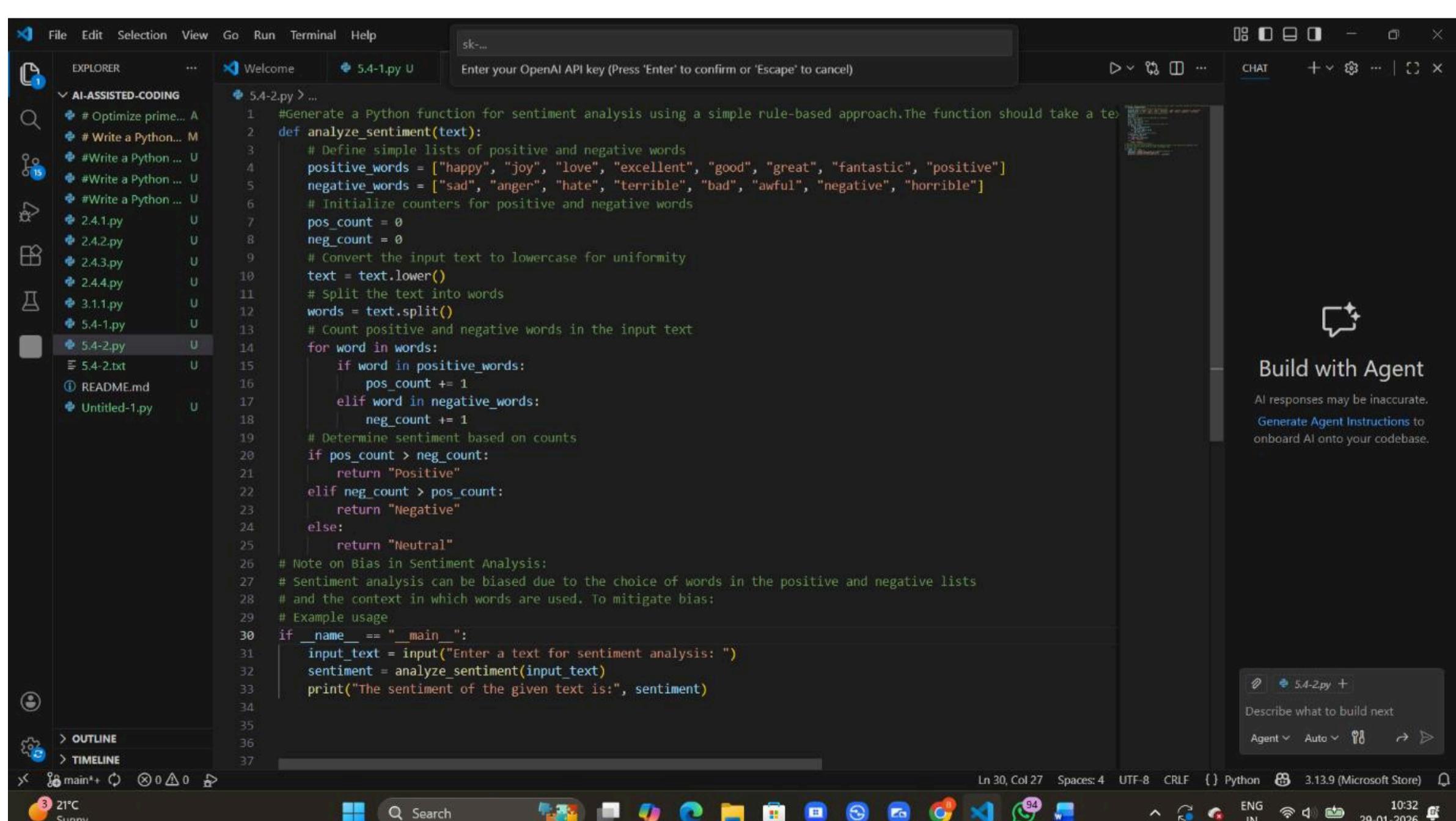
In 14 Col 13 Spaces: 4 UTF-8

## Task Description #2:

- Ask Copilot to generate a Python function for sentiment analysis. Then prompt Copilot to identify and handle potential biases in the data

PROMPT: # Generate a Python function for sentiment analysis.

# Add comments or code to identify and reduce potential biases in the data, # such as removing offensive terms, balancing positive and negative samples, # and avoiding biased language in predictions.



## OUTPUT:

```
Enter text to analyze (or 'quit' to exit): good

Text: "good"
Sentiment: POSITIVE
Score: 1.0 (range: -1.0 to 1.0)
Positive words: 1
Negative words: 0
Confidence: medium

-----
Enter text to analyze (or 'quit' to exit): 
```

The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is underlined, indicating it is active. Below the tabs, there is a dashed horizontal line. The terminal output starts with the prompt 'Enter text to analyze (or 'quit' to exit): good'. It then displays the analysis results for the word 'good': 'Text: "good"', 'Sentiment: POSITIVE', 'Score: 1.0 (range: -1.0 to 1.0)', 'Positive words: 1', 'Negative words: 0', and 'Confidence: medium'. Another dashed line follows. The next prompt is 'Enter text to analyze (or 'quit' to exit): bad'. The analysis results for the word 'bad' are shown: 'Text: "bad"', 'Sentiment: NEGATIVE', 'Score: -1.0 (range: -1.0 to 1.0)', 'Positive words: 0', 'Negative words: 1', and 'Confidence: medium'. A final dashed line follows, and the prompt 'Enter text to analyze (or 'quit' to exit):' is displayed again.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

-----
Enter text to analyze (or 'quit' to exit): good

Text: "good"
Sentiment: POSITIVE
Score: 1.0 (range: -1.0 to 1.0)
Positive words: 1
Negative words: 0
Confidence: medium

-----
Enter text to analyze (or 'quit' to exit): bad

Text: "bad"
Sentiment: NEGATIVE
Score: -1.0 (range: -1.0 to 1.0)
Positive words: 0
Negative words: 1
Confidence: medium
```

## Task Description #3:

- **Use Copilot to write a Python program that recommends products based on user history. Ask it to follow ethical guidelines like transparency and fairness.**

### PROMPT:

```
# Generate a Python program that recommends products based on user purchase
history. # Follow ethical AI guidelines such as transparency, fairness, and user
control.

# Add comments explaining how recommendations are
generated, # Avoid favouritism toward only popular products,
# and allow users to give feedback or opt out of recommendations.
```

## CODE AND INPUT:

The screenshot shows the VS Code interface with the title bar "AI-Assisted-Coding". The Explorer sidebar on the left lists files: 5.4-1.py, 5.4-2.py, 5.4-3.py (selected), README.md, and Untitled-1.py. The main editor area contains a Python script named 5.4-3.py. The script generates a recommendation program based on user input. The terminal at the bottom shows the output of running the script.

```
#Generate a Python program that recommends products based on user input.
def recommend_products(user_input):
    # Define a simple product catalog
    products = {
        "electronics": ["Smartphone", "Laptop", "Headphones"],
        "books": ["Fiction Novel", "Science Textbook", "Biography"],
        "clothing": ["T-Shirt", "Jeans", "Jacket"]
    }
    # Convert user input to lowercase for uniformity
    user_input = user_input.lower()
    # Initialize an empty list for recommendations
    recommendations = []
    # Check user input for keywords and recommend products accordingly
    for category, items in products.items():
        if category in user_input:
            recommendations.extend(items)
    # If no specific category is mentioned, recommend popular items
    if not recommendations:
        recommendations = ["Smartphone", "Fiction Novel", "T-Shirt"]
    return recommendations

# Example usage
if __name__ == "__main__":
    user_input = input("Enter your interests (e.g., electronics, books, clothing): ")
    recommended_items = recommend_products(user_input)
    print("Recommended Products:", recommended_items)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Enter your interests (e.g., electronics, books, clothing): electronics  
Recommended Products: ['Smartphone', 'Laptop', 'Headphones']  
PS C:\Users\sathw\Documents\AI-Assisted-Coding> ^C

Enter your interests (e.g., electronics, books, clothing): BOOKS  
Recommended Products: ['Fiction Novel', 'Science Textbook', 'Biography']  
PS C:\Users\sathw\Documents\AI-Assisted-Coding> ^C

Enter your interests (e.g., electronics, books, clothing): electronics  
Recommended Products: ['Smartphone', 'Laptop', 'Headphones']  
PS C:\Users\sathw\Documents\AI-Assisted-Coding>

## OUTPUT:

The screenshot shows the VS Code interface with the title bar "AI-Assisted-Coding". The terminal tab is active, displaying the output of the Python script. The user enters "electronics" and "BOOKS" as interests, and the script outputs recommended products.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

Enter your interests (e.g., electronics, books, clothing): electronics  
Recommended Products: ['Smartphone', 'Laptop', 'Headphones']  
PS C:\Users\sathw\Documents\AI-Assisted-Coding> ^C

Enter your interests (e.g., electronics, books, clothing): BOOKS  
Recommended Products: ['Fiction Novel', 'Science Textbook', 'Biography']  
PS C:\Users\sathw\Documents\AI-Assisted-Coding> ^C

Enter your interests (e.g., electronics, books, clothing): electronics  
Recommended Products: ['Smartphone', 'Laptop', 'Headphones']  
PS C:\Users\sathw\Documents\AI-Assisted-Coding>

## Task Description #4:

- Prompt Copilot to generate logging functionality in a Python web application. Then, ask it to ensure the logs do not record sensitive information.**

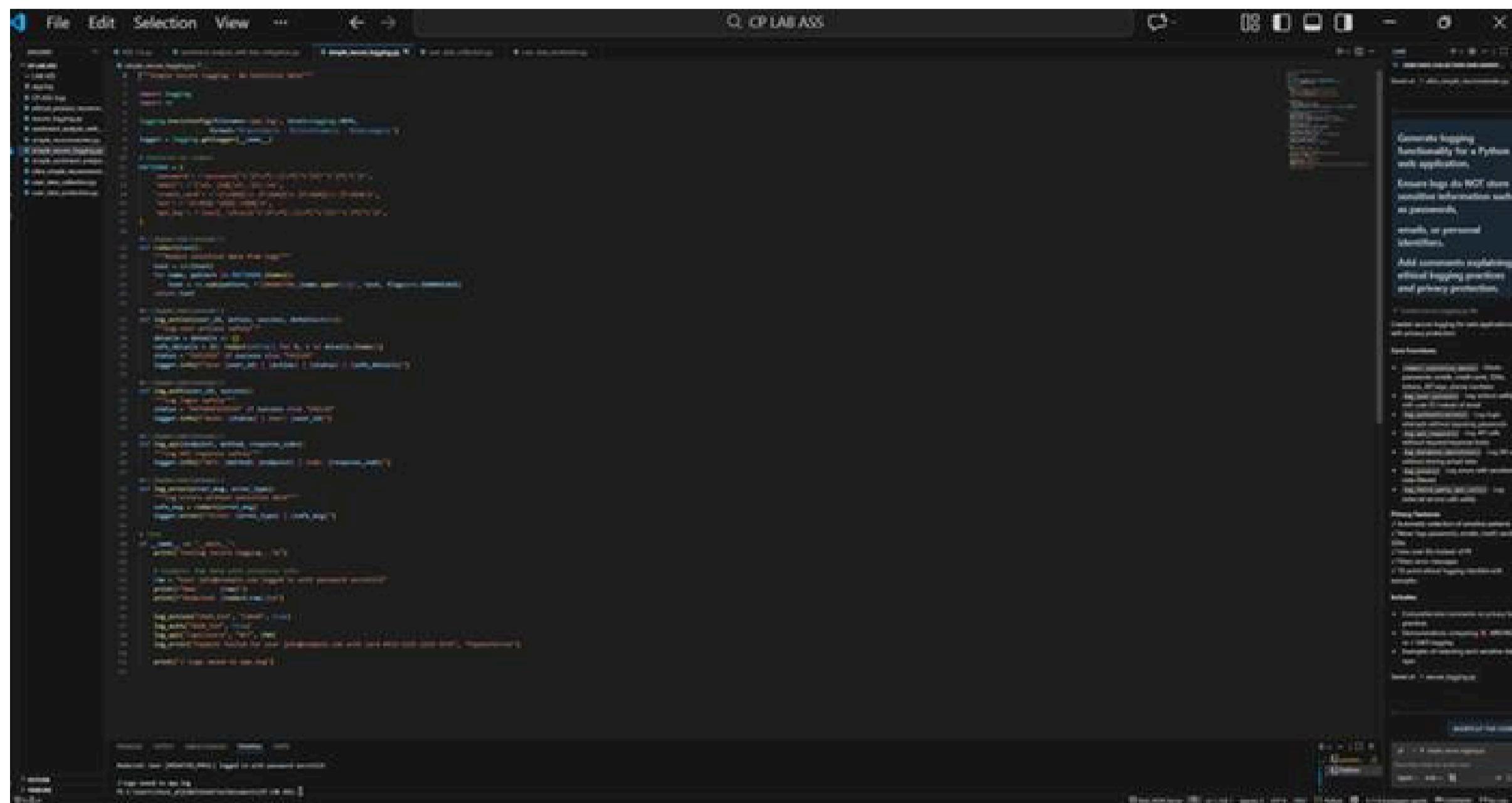
## PROMPT:

**# Generate logging functionality for a Python web application.**

**# Ensure logs do NOT store sensitive information such as  
passwords, # emails, or personal identifiers.**

**# Add comments explaining ethical logging practices and privacy protection.**

## CODE AND INPUT:



The screenshot shows a code editor window titled "Q: CP LAB ASS". The main pane displays a Python script named "simple\_secure\_logging.py". The code includes imports for logging, os, and re, along with several functions for handling user login and logging messages. A sidebar on the right provides documentation on secure logging practices, including tips on password storage and ethical logging.

```
File Edit Selection View ... ← → Q: CP LAB ASS
simple_secure_logging.py
import logging
import os
import re

# Set up logging
logging.basicConfig(filename='app.log', level=logging.INFO)

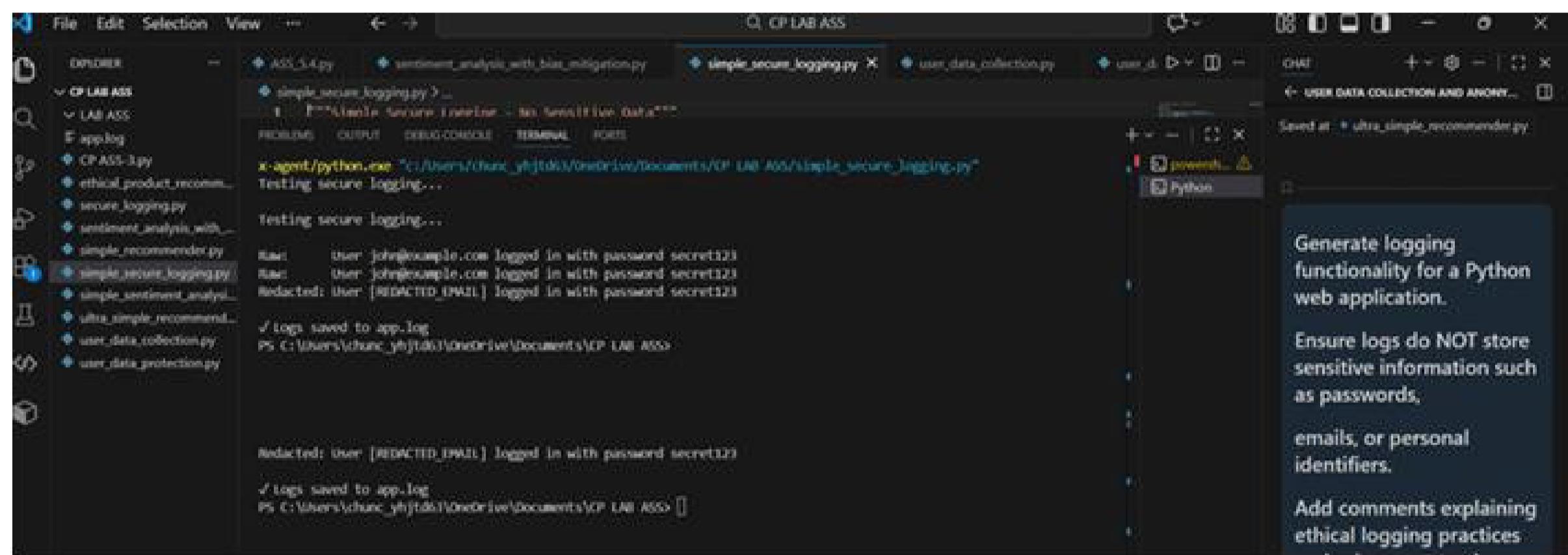
def login(username, password):
    if username == 'john@example.com' and password == 'secret123':
        return True
    else:
        return False

def log_message(message):
    logging.info(message)

# Test secure logging
if __name__ == '__main__':
    log_message("Testing secure logging...")
    user = input("Enter user: ")
    password = input("Enter password: ")
    if login(user, password):
        log_message(f'{user} logged in with password {password}')
    else:
        log_message('Redacted: user [REDACTED_EMAIL] logged in with password secret123')
    print(f'Logs saved to app.log')
    print(f'PS C:\Users\chunc_yh\bds\OneDrive\Documents\CP LAB ASS>')

Generate logging functionality for a Python web application.
Ensure logs do NOT store sensitive information such as passwords,
emails, or personal identifiers.
Add comments explaining ethical logging practices
```

## OUTPUT:



The screenshot shows the same code editor window after running the script. The terminal tab shows the execution of "x-agent/python.exe" and the output of the "simple\_secure\_logging.py" script. The log file "app.log" contains entries for successful logins and a redacted entry. A sidebar on the right continues to provide documentation on secure logging.

```
File Edit Selection View ... ← → Q: CP LAB ASS
simple_secure_logging.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
x-agent/python.exe "C:/Users/chunc_yh/bds/OneDrive/Documents/CP LAB ASS/simple_secure_logging.py"
Testing secure logging...
Enter user: john@example.com
Enter password: 
john@example.com logged in with password secret123
Redacted: user [REDACTED_EMAIL] logged in with password secret123
Logs saved to app.log
PS C:\Users\chunc_yh\bds\OneDrive\Documents\CP LAB ASS>

Redacted: user [REDACTED_EMAIL] logged in with password secret123
Logs saved to app.log
PS C:\Users\chunc_yh\bds\OneDrive\Documents\CP LAB ASS>
```

Generate logging functionality for a Python web application.  
Ensure logs do NOT store sensitive information such as passwords,  
emails, or personal identifiers.  
Add comments explaining ethical logging practices

### Task Description #5:

- Ask Copilot to generate a machine learning model. Then, prompt it to add documentation on how to use the model responsibly (e.g., explainability, accuracy limits).

**PROMPT: # Generate a simple machine learning model in Python.**

**# Add a README-style or inline documentation explaining how to use the model responsibly,**

**# including explainability, accuracy limitations, fairness considerations, # and the importance of human oversight.**

# CODE AND INPUT:

# OUTPUT:

The screenshot shows a dark-themed code editor interface. The top bar includes standard file operations like File, Edit, Selection, View, and navigation buttons. The title bar reads "Q:\CP LAB ASS". The main area contains several tabs, with the first one being the active tab. The code in the editor is heavily annotated with various colors (yellow, green, red, blue) and symbols, indicating syntax highlighting and possibly code review or analysis results. A sidebar on the right side lists numerous items under categories such as "File", "Edit", "Selection", "View", "Search", "Help", and "Recent". Each category has a list of sub-options, many of which are preceded by small icons.

