

AI Assisted Coding LAB ASS-5.4

NAME: B. SATHWIK REDDY

BATCH:14

2303A510B1

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:

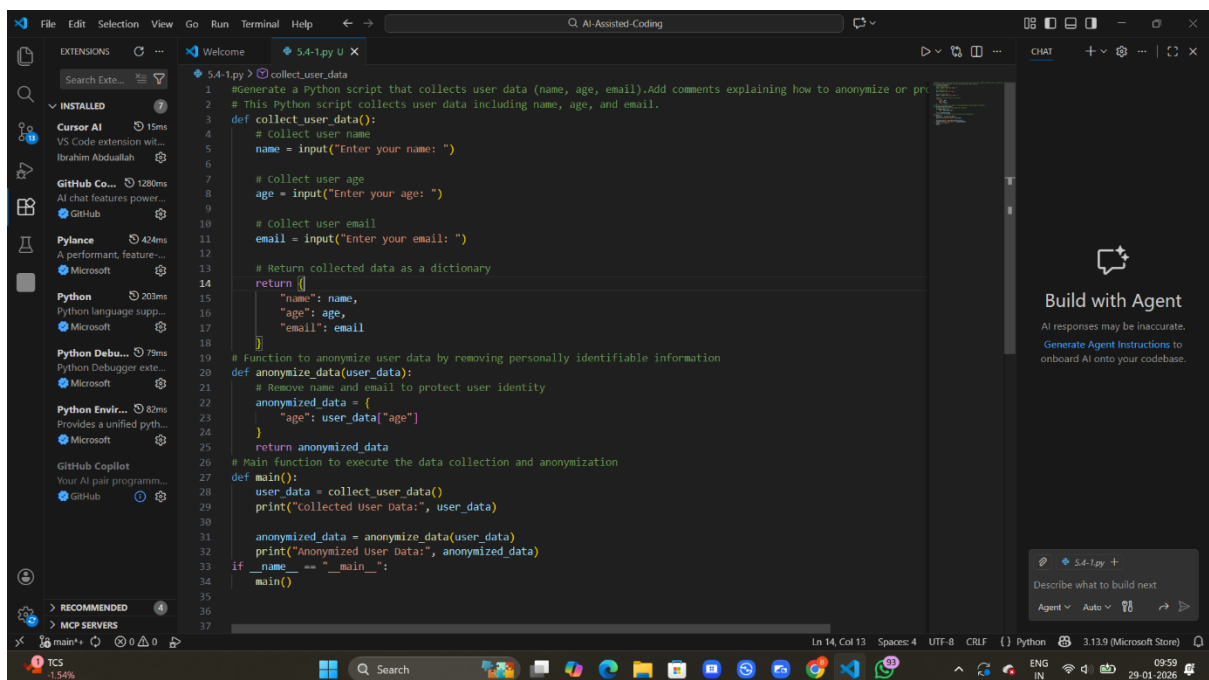
Generate a Python script that collects user data such as name, age, and email.

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



```
1 generate a python script that collects user data (name, age, email).Add comments explaining how to anonymize or pr
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()
```

OUTPUT:

```
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
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>
```

Task Description #2:

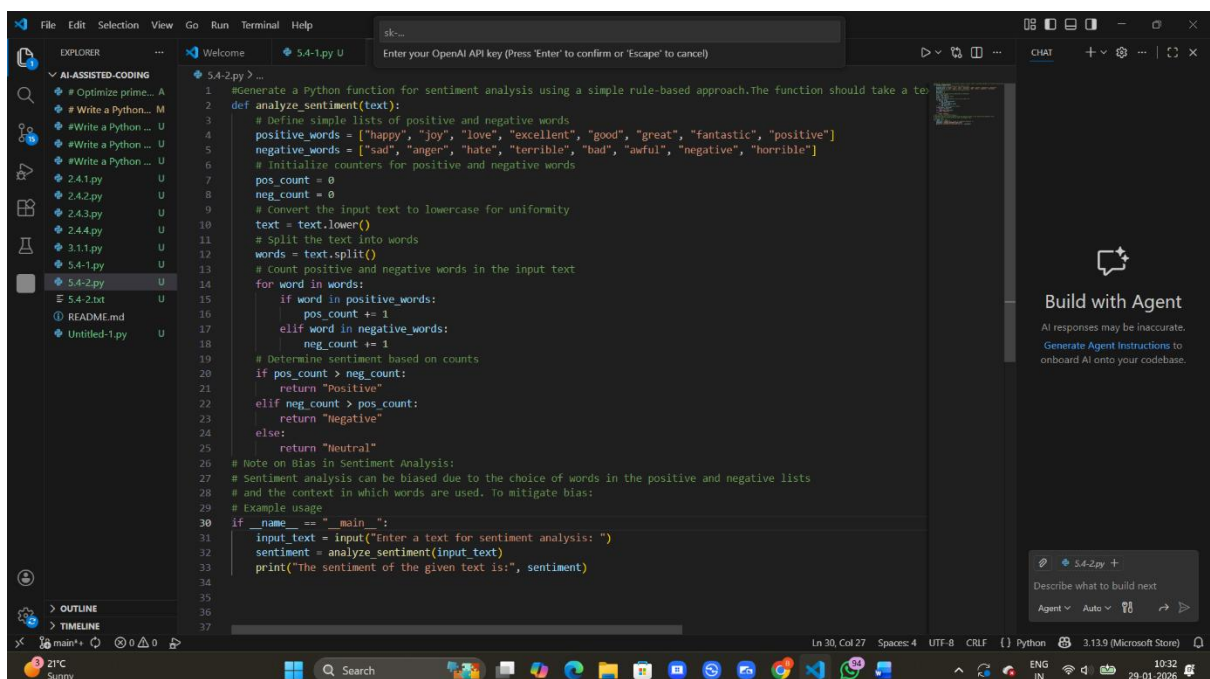
• Ask Copilot to generate a Python function for sentiment analysis. Then prompt Copilot to identify and handle potential biases in the data

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.



```
File Edit Selection View Go Run Terminal Help
Welcome S4-1.py U
Enter your OpenAI API key (Press 'Enter' to confirm or 'Escape' to cancel)
CHAT + - - - - -
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
S4-2.py +
Describe what to build next
Agent Auto 10:32 29.01.2026

5.4-2.py
1 #Generate a Python function for sentiment analysis using a simple rule-based approach.The function should take a text
2 def analyze_sentiment(text):
3     # Define simple lists of positive and negative words
4     positive_words = ["happy", "joy", "love", "excellent", "good", "great", "fantastic", "positive"]
5     negative_words = ["sad", "anger", "hate", "terrible", "bad", "awful", "negative", "horrible"]
6     # Initialize counters for positive and negative words
7     pos_count = 0
8     neg_count = 0
9     # Convert the input text to lowercase for uniformity
10    text = text.lower()
11    # Split the text into words
12    words = text.split()
13    # Count positive and negative words in the input text
14    for word in words:
15        if word in positive_words:
16            pos_count += 1
17        elif word in negative_words:
18            neg_count += 1
19    # Determine sentiment based on counts
20    if pos_count > neg_count:
21        return "Positive"
22    elif neg_count > pos_count:
23        return "Negative"
24    else:
25        return "Neutral"
26    # Note on Bias in Sentiment Analysis:
27    # Sentiment analysis can be biased due to the choice of words in the positive and negative lists
28    # and the context in which words are used. To mitigate bias:
29    # Example usage
30    if __name__ == "__main__":
31        input_text = input("Enter a text for sentiment analysis: ")
32        sentiment = analyze_sentiment(input_text)
33        print("The sentiment of the given text is:", sentiment)
34
35
36
37
```

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): 
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

-----

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
-----

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

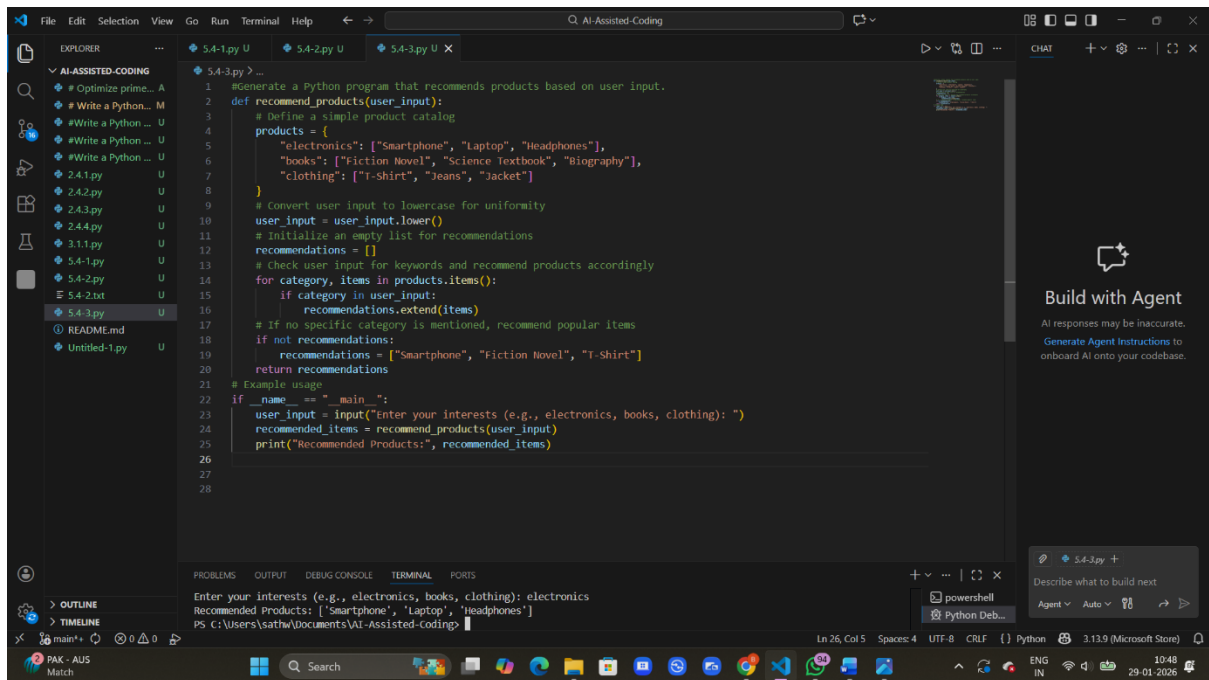
Task Description #3:

- 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:

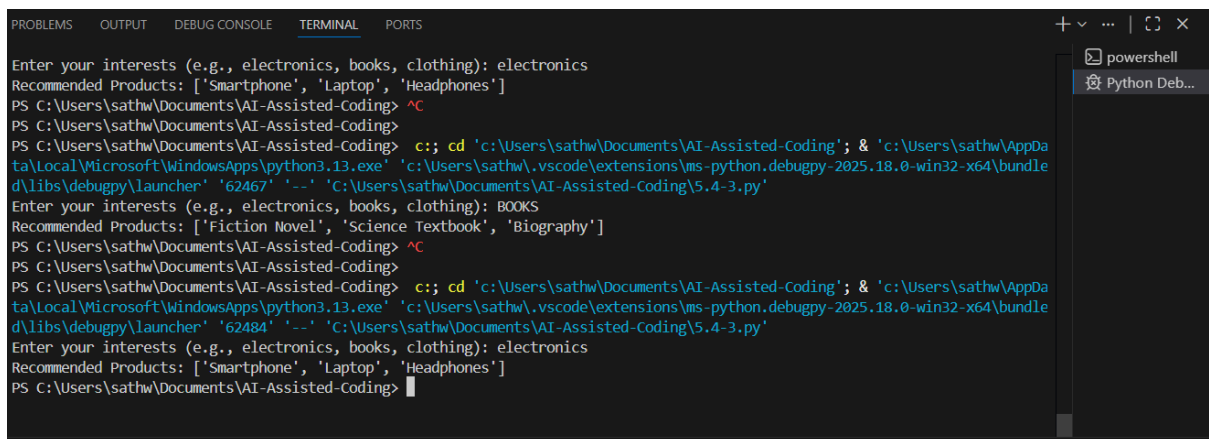


The screenshot shows the Visual Studio Code editor with a Python file named `5.4-3.py`. The code defines a `recommend_products` function that takes `user_input` and returns a list of recommended products based on keywords like 'electronics', 'books', and 'clothing'. The terminal shows the execution of the script, which prompts the user for interests and outputs the recommended products. On the right, a chat window titled 'Build with Agent' is visible, showing a conversation with an AI agent.

```
1 #Generate a Python program that recommends products based on user input.
2 def recommend_products(user_input):
3     # Define a simple product catalog
4     products = {
5         "electronics": ["Smartphone", "Laptop", "Headphones"],
6         "books": ["Fiction Novel", "Science Textbook", "Biography"],
7         "clothing": ["T-Shirt", "Jeans", "Jacket"]
8     }
9     # Convert user input to lowercase for uniformity
10    user_input = user_input.lower()
11    # Initialize an empty list for recommendations
12    recommendations = []
13    # Check user input for keywords and recommend products accordingly
14    for category, items in products.items():
15        if category in user_input:
16            recommendations.extend(items)
17    # If no specific category is mentioned, recommend popular items
18    if not recommendations:
19        recommendations = ["Smartphone", "Fiction Novel", "T-Shirt"]
20    return recommendations
21 # Example usage
22 if __name__ == "__main__":
23     user_input = input("Enter your interests (e.g., electronics, books, clothing): ")
24     recommended_items = recommend_products(user_input)
25     print("Recommended Products:", recommended_items)
26
27
28
```

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 terminal output of the Python script. It displays the prompt for user interests, the resulting recommended products, and the execution of the script in a PowerShell environment.

```
Enter your interests (e.g., electronics, books, clothing): electronics
Recommended Products: ['Smartphone', 'Laptop', 'Headphones']
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
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' '62467' '--' 'c:\Users\sathw\Documents\AI-Assisted-Coding\5.4-3.py'
Enter your interests (e.g., electronics, books, clothing): BOOKS
Recommended Products: ['Fiction Novel', 'Science Textbook', 'Biography']
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
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' '62484' '--' 'c:\Users\sathw\Documents\AI-Assisted-Coding\5.4-3.py'
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 with a file explorer on the left and a search bar at the top. The main editor area displays the code for `simple_secure_logging.py`. The code includes imports for `logging`, `os`, and `sys`. It defines a `Logger` class with methods for logging user login attempts, including password hashing and redaction. The code also includes a `main` function for testing the logging functionality. The right sidebar shows a search results panel with the file `simple_secure_logging.py` selected.

OUTPUT:

The screenshot shows a terminal window with the command `x-agent/python.exe "c:\Users\chun_yh\OneDrive\Documents\CP LAB ASS\simple_secure_logging.py"` executed. The output shows the following:

```
Testing secure logging...
Testing secure logging...
Raw: user johng@example.com logged in with password secret123
Raw: user johng@example.com logged in with password secret123
Redacted: user [REDACTED_9NA11] logged in with password secret123
✓ logs saved to app.log
PS C:\Users\chun_yh\OneDrive\Documents\CP LAB ASS>

Redacted: user [REDACTED_9NA11] logged in with password secret123
✓ logs saved to app.log
PS C:\Users\chun_yh\OneDrive\Documents\CP LAB ASS>
```

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).

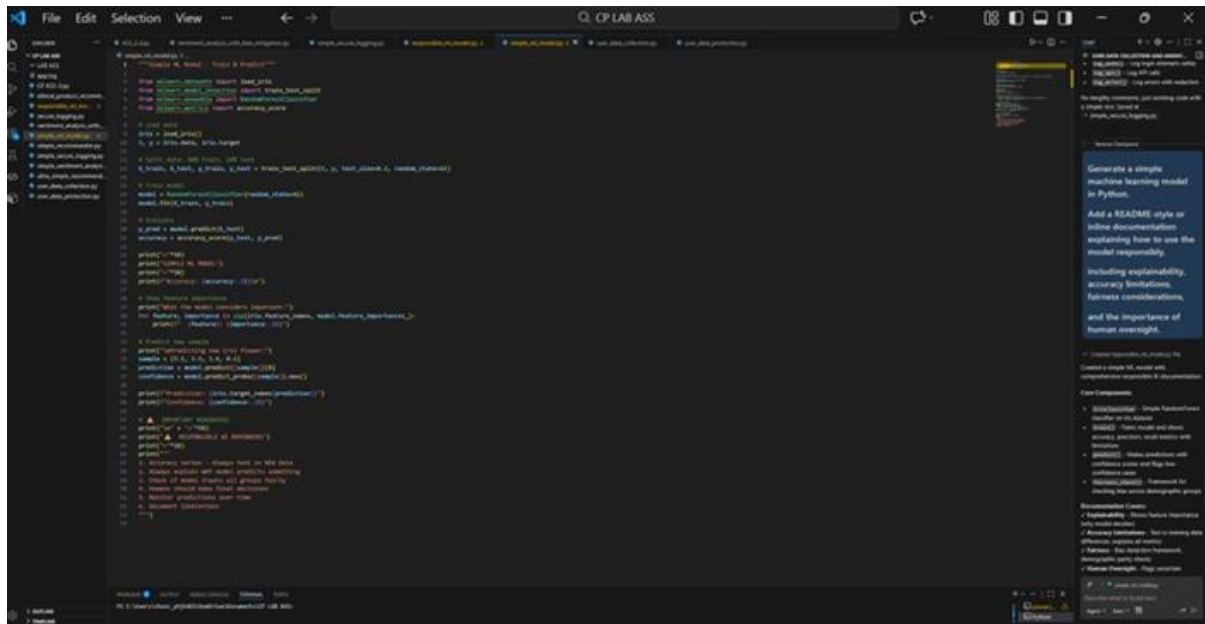
PROMPT: # Generate a simple machine learning model in Python.

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:

