

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:

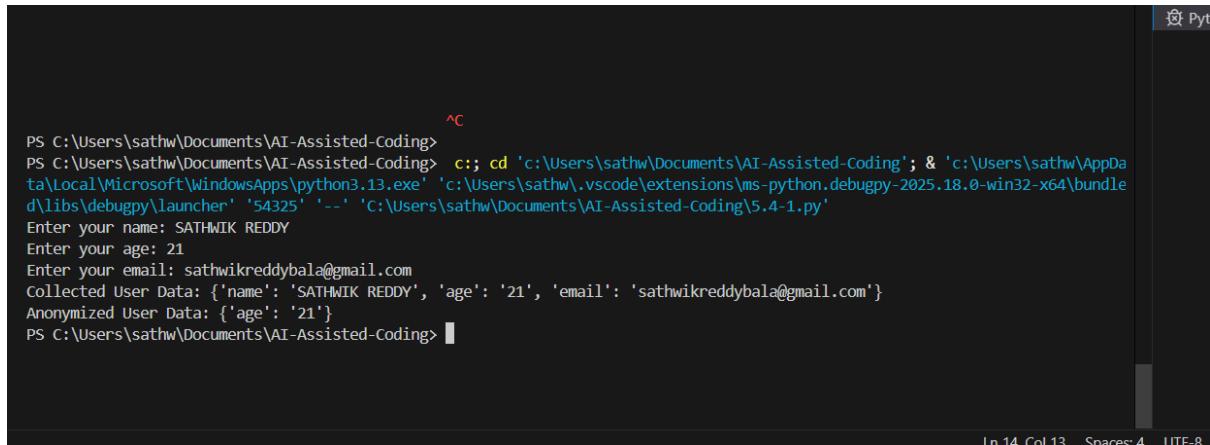
```
# Generate a Python script that collects user data such as name, age, and email.  
  
# Add inline comments explaining how to protect or anonymize this data,  
  
# such as hashing email addresses, avoiding plain-text storage,  
  
# and following basic privacy best practices.
```

CODE AND INPUT

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

- File Explorer:** Shows extensions installed: Cursor AI (15ms), GitHub Co... (1280ms), Pylance (424ms), Python (203ms), Python Debugger (79ms), Python Envir... (82ms), and GitHub Copilot.
- Code Editor:** A Python script named "5.4-1.py" is open, titled "Welcome". The code collects user data (name, age, email) and anonymizes it by removing personally identifiable information (name and email). It includes a main function that prints the collected and anonymized data.
- Bottom Status Bar:** Shows file path (5.4-1.py), line 14, column 13, spaces 4, UTF-8, CRLF, Python 3.13.9 (Microsoft Store), and a timestamp (09:59 AM IN 29-01-2026).
- Bottom Taskbar:** Includes icons for File, Edit, Selection, View, Go, Run, Terminal, Help, Search, and various system icons.
- Right Sidebar:** Features a "CHAT" tab with a message icon, a "Build with Agent" section with a note about AI responses being inaccurate and instructions to onboard AI, and a "Describe what to build next" input field.

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' '5425' '--' 'c:\Users\sathw\Documents\AI-Assisted-Coding\5.4-1.py'
Enter your name: SATHWIK REDDY
Enter your age: 21
Enter your email: sathwikreddybala@gmail.com
Collected User Data: {'name': 'SATHWIK REDDY', 'age': '21', 'email': 'sathwikreddybala@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:

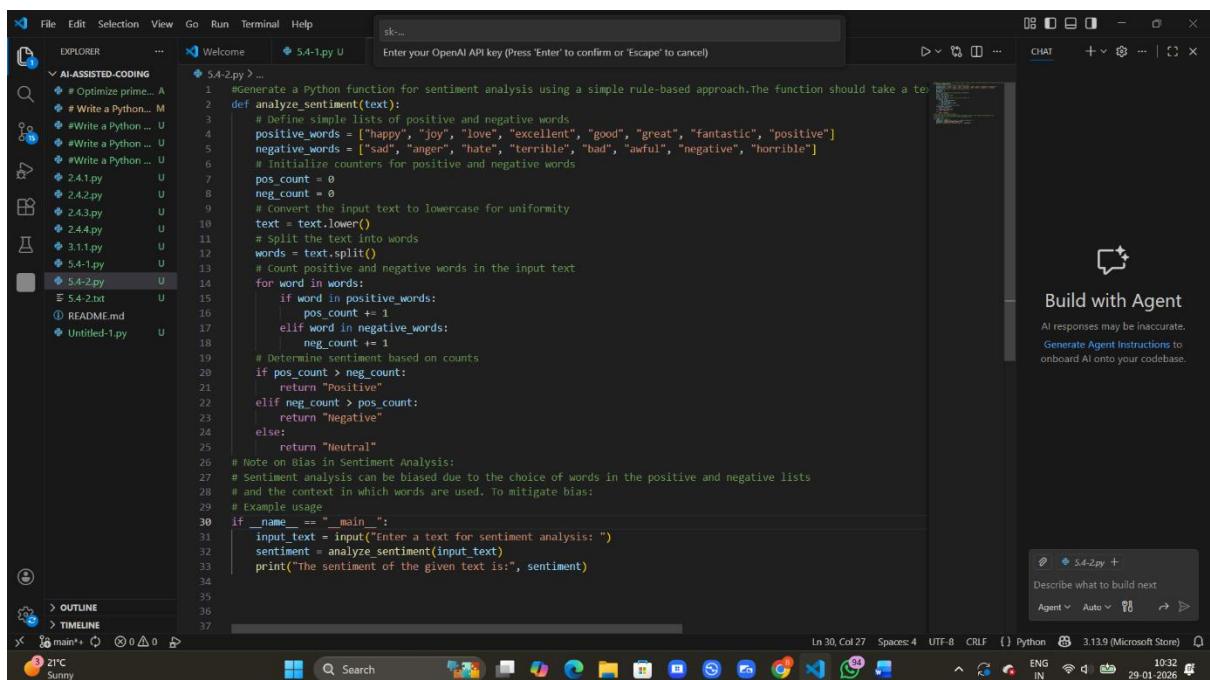
- Ask Copilot to generate a Python function for sentiment analysis. Then prompt Copilot to identify and handle potential biases in the data

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.



```
File Edit Selection View Go Run Terminal Help sk...
EXPLORER Welcome 5.4-1.py U Enter your OpenAI API key (Press 'Enter' to confirm or 'Escape' to cancel)
AI-ASSISTED-CODING
# Optimize prime... A
# Write a Python... M
#Write a Python ... U
#Write a Python ... U
#Write a Python ... U
2.4.1.py U
2.4.2.py U
2.4.3.py U
2.4.4.py U
3.1.1.py U
5.4-1.py U
5.4-2.py U
5.4-2.txt U
README.md
Untitled-1.py
5.4-2.py U
# Generate a Python function for sentiment analysis using a simple rule-based approach. The function should take a text input and return the sentiment (Positive, Negative, or Neutral).
# Define simple lists of positive and negative words
positive_words = ["happy", "joy", "love", "excellent", "good", "great", "fantastic", "positive"]
negative_words = ["sad", "anger", "hate", "terrible", "bad", "awful", "negative", "horrible"]
# Initialize counters for positive and negative words
pos_count = 0
neg_count = 0
# Convert the input text to lowercase for uniformity
text = text.lower()
# Split the text into words
words = text.split()
# Count positive and negative words in the input text
for word in words:
    if word in positive_words:
        pos_count += 1
    elif word in negative_words:
        neg_count += 1
# Determine sentiment based on counts
if pos_count > neg_count:
    return "Positive"
elif neg_count > pos_count:
    return "Negative"
else:
    return "Neutral"
# Note on Bias in Sentiment Analysis:
# Sentiment analysis can be biased due to the choice of words in the positive and negative lists
# and the context in which words are used. To mitigate bias:
# Example usage:
if __name__ == "__main__":
    input_text = input("Enter a text for sentiment analysis: ")
    sentiment = analyze_sentiment(input_text)
    print("The sentiment of the given text is:", sentiment)
```

CHAT + ... | x

Build with Agent

All responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

Describe what to build next

Agent Auto ▾

In 30, Col 27, Spaces: 4, UTF-8, CR/LF, Python, 3.13.9 (Microsoft Store), ENG IN, 10:32, 29-01-2026

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 several tabs at the top: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The main area displays two sets of text analysis results separated by a dashed line.

```
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
```

```
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:

```
# Generate a Python program that recommends products based on 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)
```

OUTPUT:

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

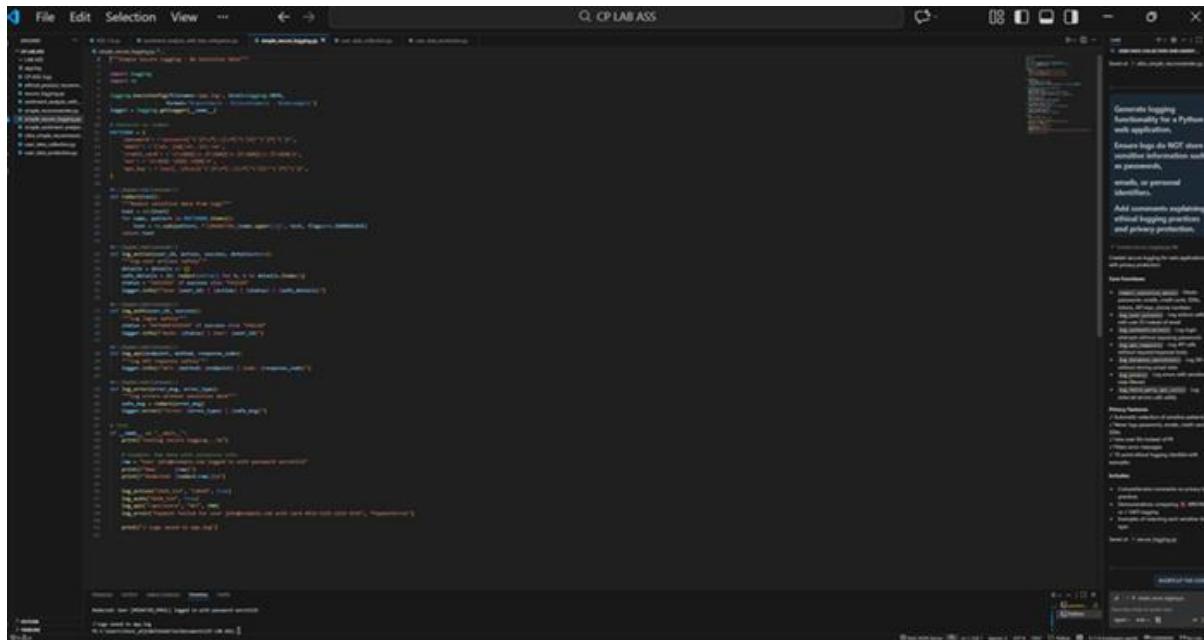
PROMPT:

```
# 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 VS Code interface with multiple tabs open. The main editor tab contains Python code for generating secure logging. A sidebar on the right provides documentation and best practices for ethical logging.

```
def generate_logging_functionality():
    # 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 considerations.

    # Create logger object
    logger = logging.getLogger(__name__)
    logger.setLevel(logging.INFO)

    # Create file handler
    file_handler = logging.FileHandler('app.log')
    file_handler.setLevel(logging.INFO)

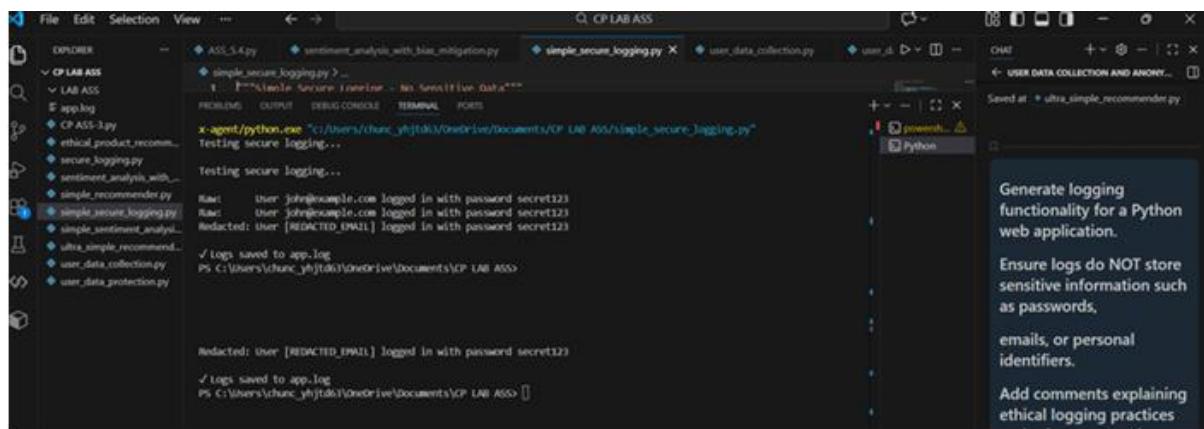
    # Create formatter
    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')

    # Add formatter to file handler
    file_handler.setFormatter(formatter)

    # Add file handler to logger
    logger.addHandler(file_handler)

    # Test logging
    logger.info("User logged in with password: secret123")
    logger.info("User [REDACTED_EMAIL] logged in with password: secret123")
    logger.info("Logs saved to app.log")
```

OUTPUT:



The screenshot shows the VS Code interface with the terminal tab active. The user has run the script and is viewing the log output.

```
x-agent/python.exe "C:\Users\chunc_yh\Downloads\Documents\CP LAB ASS\simple_secure_logging.py"
Testing secure Logging...
Testing secure Logging...
[REDACTED: user [REDACTED_EMAIL] logged in with password secret123]
[REDACTED: user [REDACTED_EMAIL] logged in with password secret123]
Logs saved to app.log
PS C:\Users\chunc_yh\Downloads\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:

The screenshot shows a Jupyter Notebook environment with the following details:

- File Bar:** File, Edit, Selection, View, Help.
- Title Bar:** CP LAB ASS
- Code Cell:** The cell contains Python code for a machine learning model. It includes imports for numpy, pandas, and various scikit-learn modules like LogisticRegression, GridSearchCV, and Pipeline. The code defines a pipeline for feature selection, logistic regression, and cross-validation. It also includes a function to calculate feature importance and a final step for generating a confusion matrix.
- Sidebar:** A sidebar on the right provides AI-generated documentation for the code. It includes:
 - General:** "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 Components:**
 - ML Pipeline:** Single function to handle all steps.
 - Feature Selection:** Selects most relevant features.
 - Model Training:** Trains a logistic regression model.
 - Confusion Matrix:** Confusion matrix with confidence scores and diag line.
 - Classification Report:** Classification report for each class.
 - Documentation Checks:**
 - ✓ **Documentation Quality:** Overall document quality is good.
 - ✓ **Accuracy Verification:** No error in running code.
 - ✓ **Performance:** Performance is acceptable.
 - ✓ **Fairness:** Fairness analysis is present.
 - ✓ **Explainability:** Explainability is present.
- Bottom Bar:** Includes tabs for Home, Recent, and Help.

OUTPUT:

The screenshot shows a Java IDE interface with a large codebase displayed in the center. The code is heavily annotated with red and yellow highlights, indicating numerous errors and warnings. A vertical scroll bar is visible on the right side of the code editor. On the far right, there is a detailed error list and a 'Search' bar. The top navigation bar includes 'File', 'Edit', 'Selection', 'View', and 'Help'.

