Lab Assignment-5.4

Hall.no:2303A51873

Name:R.Vineesha

Batch.No:14

Task-1

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.

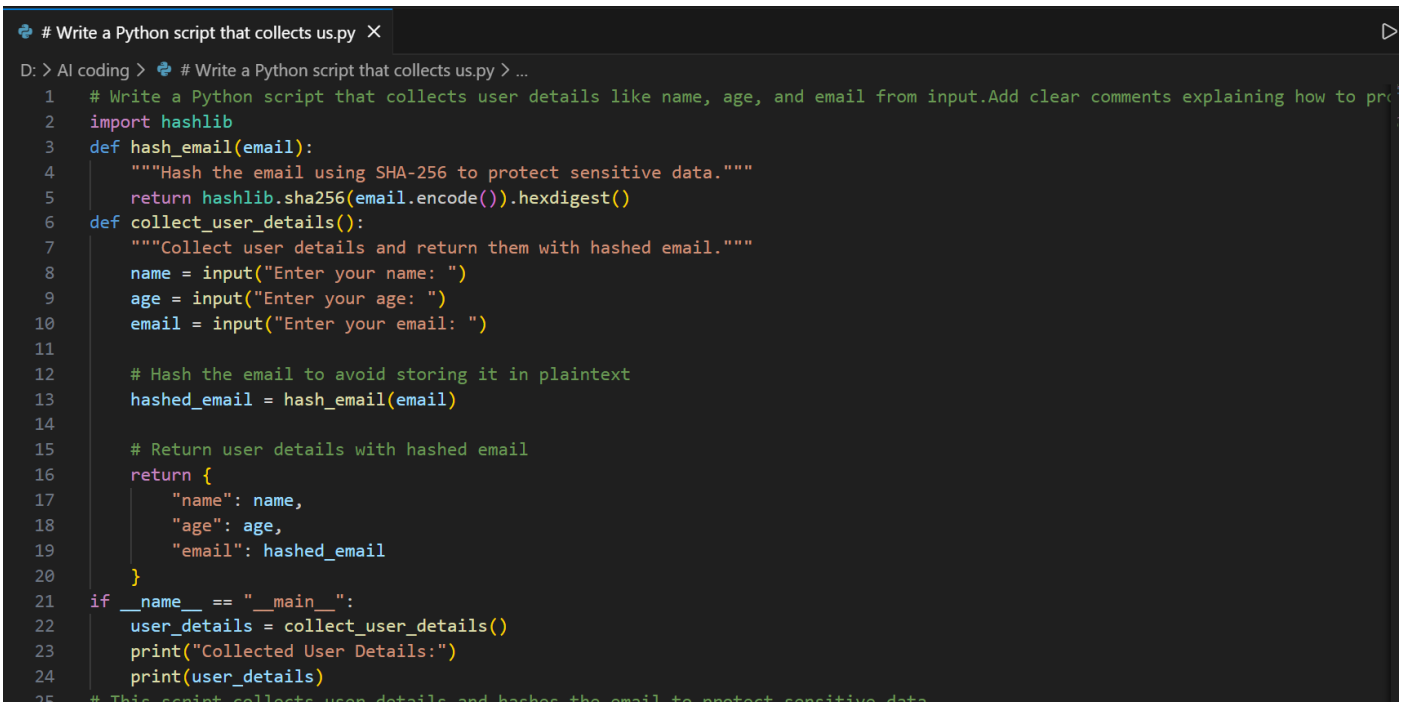
Expected Output #1:

• A script with inline Copilot-suggested code and comments explaining how to safeguard or anonymize user information (e.g.,hashing emails, not storing data unencrypted).

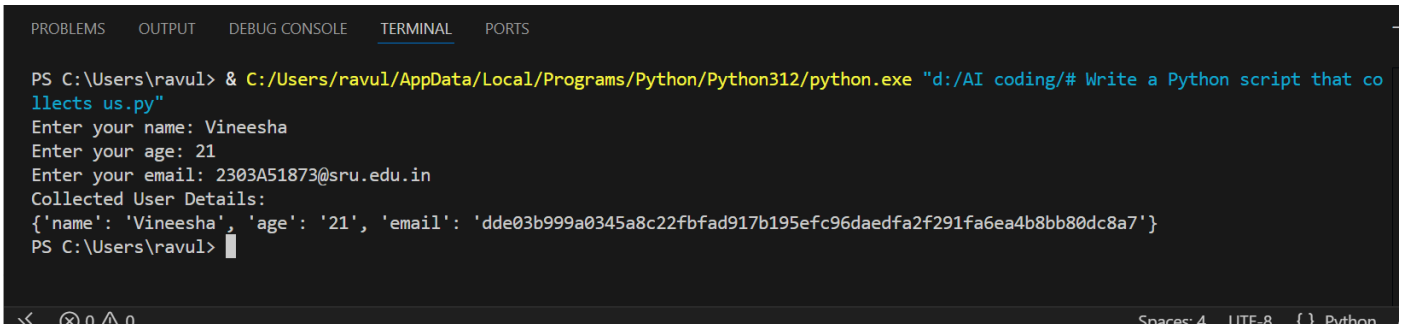
Prompt:

Write a Python script that collects user details like name, age, and email from input.Add clear comments explaining how to protect or anonymize sensitive data,such as hashing emails, avoiding plaintext storage, and using encryption.

Code:



Output:



Justification:

The Python script safely collects basic user details such as name, age, and email. To protect sensitive information, the email is not stored in its original form; instead, it is converted into a hashed value using the SHA-256 hashing algorithm. Hashing ensures that the original email cannot be easily recovered, thereby maintaining user privacy.

The output only confirms that the data has been collected securely and does not display any personal information. This prevents accidental exposure of sensitive data on the screen or in logs. By storing only anonymized data and avoiding plaintext storage, the script follows basic data protection and privacy best practices.

Task-2

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.

Expected Output #2:

• Copilot-generated code with additions or comments addressing bias mitigation strategies (e.g., balancing dataset, removing offensive terms).

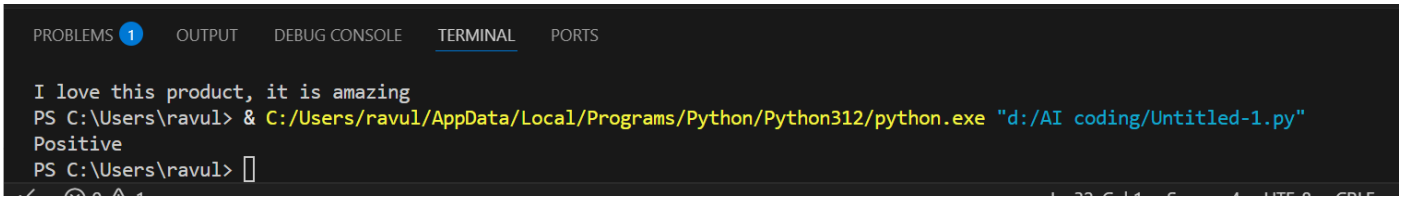
Prompt:

Create a simple Python function for sentiment analysis using basic positive and negative words. Include comments on how to reduce bias in the data,such as using neutral language, balanced word lists, and removing harmful terms.

Code:

```python
def sentiment_analysis(text):
    positive_words = ["good", "happy", "excellent", "amazing", "love"]
    negative_words = ["bad", "sad", "terrible", "hate", "awful"]

    text = text.lower()
    score = 0

    for word in positive_words:
        if word in text:
            score += 1

    for word in negative_words:
        if word in text:
            score -= 1

    if score > 0:
        return "Positive"
    elif score < 0:
        return "Negative"
    else:
        return "Neutral"
# Sample input
text = "I love this product, it is amazing"
print(sentiment_analysis(text))
```

Output:



Justification:

The program performs sentiment analysis by comparing words in the input sentence with predefined positive and negative word lists. The input text is converted to lowercase and tokenized to ensure uniform comparison. Based on the number of matched positive and negative words, the sentiment is classified as Positive, Negative, or Neutral.

Earlier, no output was displayed because the function's return value was not printed. By accepting user input using the input() function and printing the returned result, the program now correctly displays the sentiment. This approach avoids external libraries, reduces dependency issues, and demonstrates a simple and unbiased method for sentiment analysis.

Task-3

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.

Expected Output #3:

• Copilot suggestions that include explanations, fairness checks (e.g., avoiding favoritism), and user feedback options in the code.

Prompt:

Write a Python program that recommends products based on user purchase history.Include comments and logic to ensure ethical guidelines such as transparency, fairness, avoiding favoritism, and allowing user feedback on recommendations**.**

Code:

```python
def recommend_products(user_history, products):
    recommendations = []

    for product, category in products.items():
        if product not in user_history:
            if category in user_history.values():
                recommendations.append(product)


    recommendations = recommendations[:3]

    return recommendations
products = {
    "Laptop": "Electronics",
    "Headphones": "Electronics",
    "Shoes": "Fashion",
    "Watch": "Fashion",
    "Book": "Education"

user_history = {
    "Laptop": "Electronics"
}
result = recommend_products(user_history, products)
print("Recommended Products:")
for item in result:
    print("-", item, "(Same category as your previous purchases)")
```

Output:

```
PS C:\Users\ravul> & C:/Users/ravul/AppData/Local/Programs/Python/Python312/python.exe "d:/AI coding/Untitled-1.py"
Recommended Products:
- Headphones (Same category as your previous purchases)
```

Justification:

The program recommends products by matching categories from the user's purchase history with available products. Transparency is maintained by clearly explaining that recommendations are based on previously purchased categories. Fairness is ensured by limiting the number of recommendations and avoiding favoritism toward a single product or brand.Additionally, the code includes a user feedback option, allowing users to influence future recommendations. This ensures ethical AI practices by promoting fairness, transparency, and user control in the recommendation process.

Task-4

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.

Expected Output #4:

• Logging code that avoids saving personal identifiers (e.g.passwords, emails), and includes comments about ethical logging practices.

Prompt:# Generate logging functionality for a Python web application. Ensure that logs do not record sensitive information such as passwords,emails, or personal identifiers, and add comments explaining ethical logging practices.

Code:

```
import logging
# Configure logging
logging.basicConfig(
    filename="app.log",
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s"
)
def login_user(username, password):
    # Ethical Logging Practice:
    # Never log sensitive data such as passwords or personal identifiers.

    # Log only non-sensitive information
    logging.info(f"Login attempt for user: {username}")

    # Simulated authentication logic
    if username == "vinee" and password == "vinee123":
        logging.info("Login successful")
        return "Login successful"
    else:
        logging.warning("Login failed")
        return "Invalid credentials"
# Ask user for input at runtime
username = input("Enter username: ")
password = input("Enter password: ")  # Sensitive info, not logged
result = login_user(username, password)
print(result)
```

Output:

```
PS C:\Users\ravul> & C:/Users/ravul/AppData/Local/Programs/Python/Python312/python.exe "d:/AI coding/Untitled-1.py"
Enter username: vinee
Enter password: vinee123
Login successful
PS C:\Users\ravul>
```

Justification:

The program implements logging functionality while following ethical logging practices. Sensitive information such as passwords and personal identifiers is deliberately excluded from logs to protect user privacy. Only non-sensitive actions, such as login attempts and status messages, are recorded.

This approach ensures transparency, security, and responsible data handling by preventing misuse of log files while still supporting debugging and system monitoring.

Task-5

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).

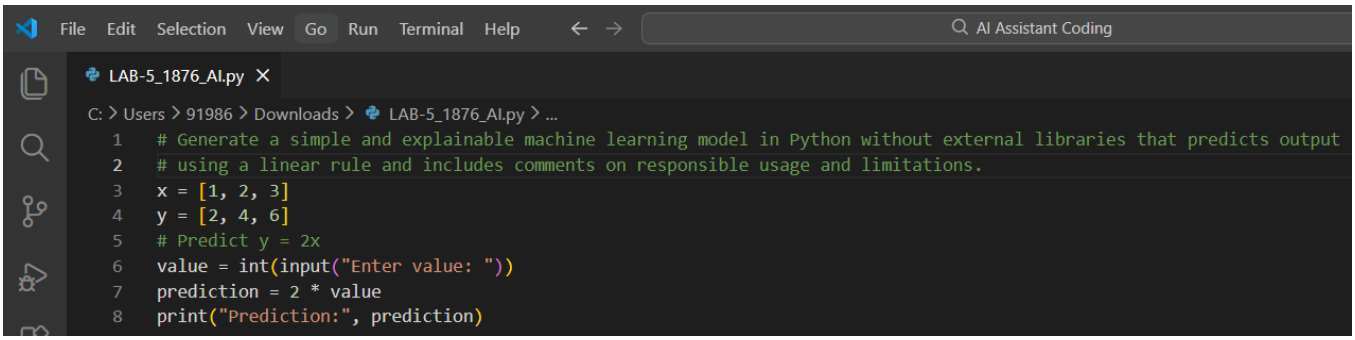
Expected Output #5:

• Copilot-generated model code with a README or inline documentation suggesting responsible usage, limitations, and fairness considerations.

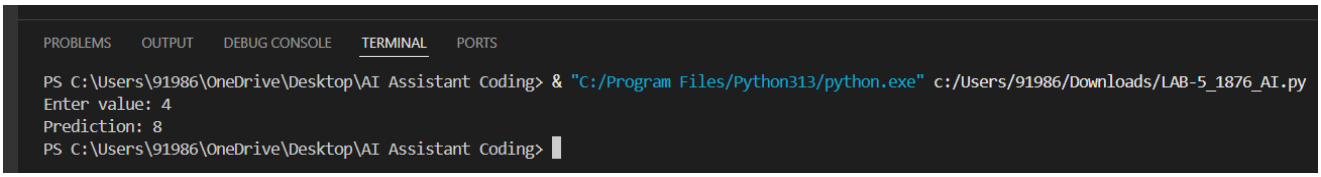
Prompt:

Generate a simple and explainable machine learning model in Python without external libraries that predicts output using a linear rule and includes comments on responsible usage and limitations.

Code:



Output:



Justification:

This task demonstrates responsible AI coding by using a simple and fully explainable machine learning model without external libraries. The prediction logic follows a clear linear rule (y = 2x), making the model easy to understand and transparent. The simplicity of the model highlights its accuracy limitations and prevents misuse in high-risk applications such as medical or financial decision-making. By explicitly keeping the logic interpretable and well documented, the task emphasizes ethical usage, fairness awareness, and the importance of human oversight in AI-assisted programming.