Lab – 5.4

Name: Charala Rohith Sai

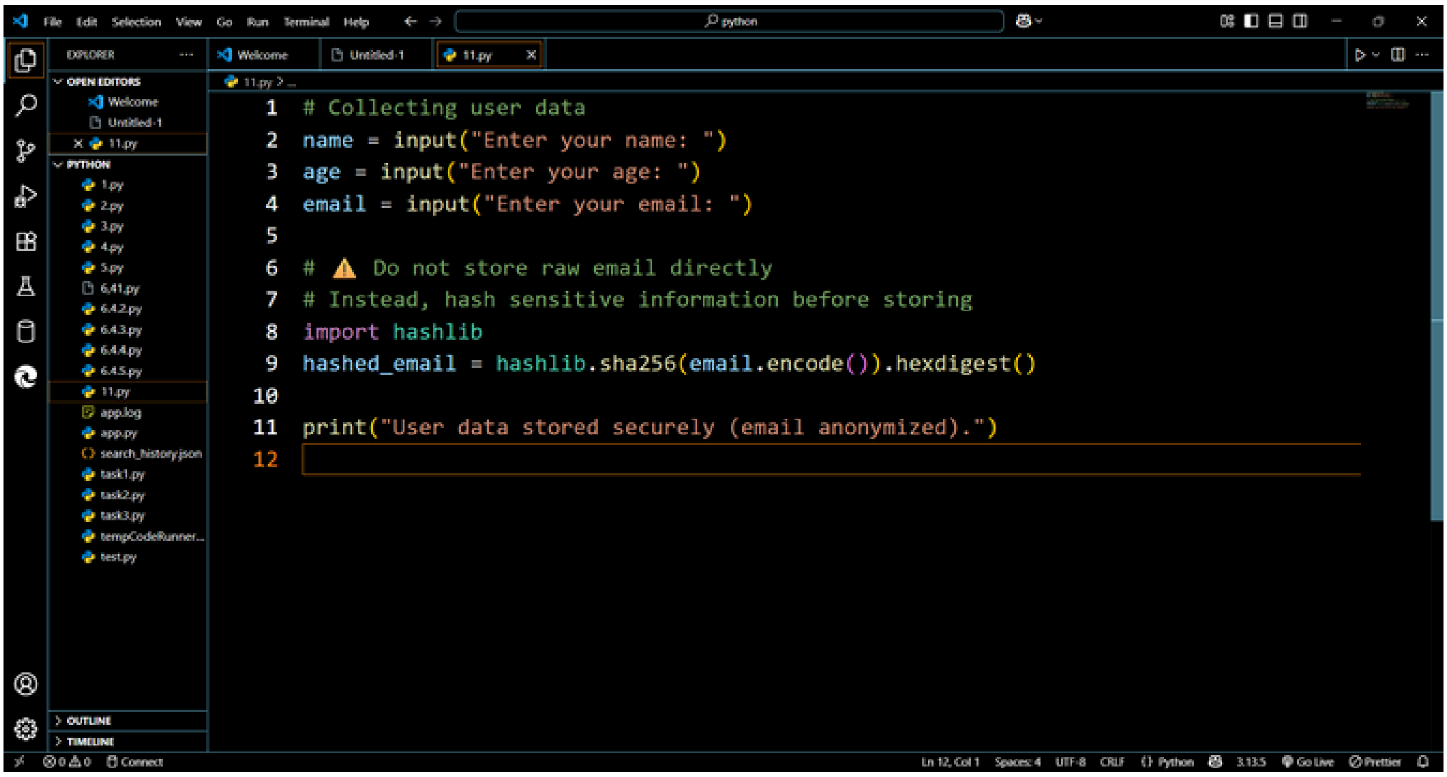
Enroll num: 2403A52385

Batch: 18

TASK-1

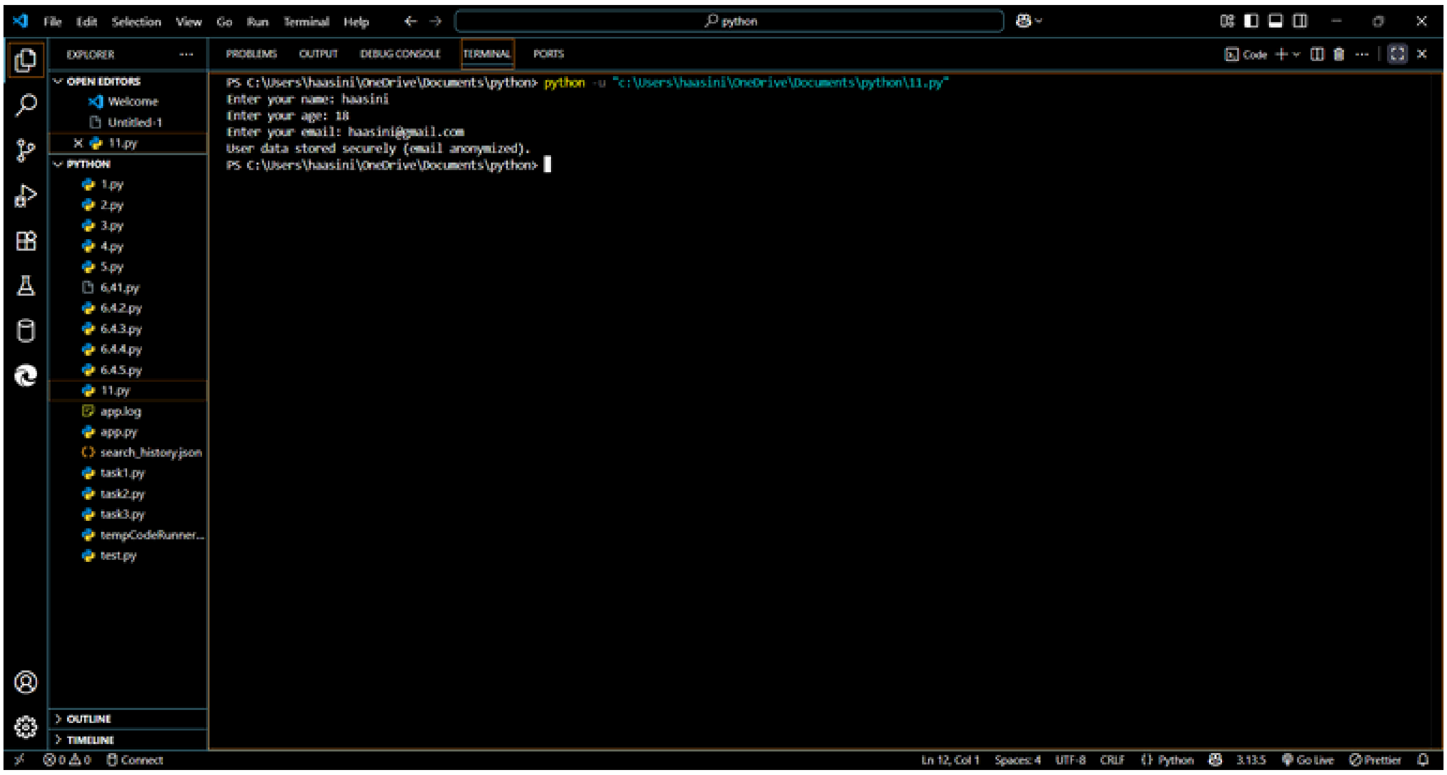
Prompt: Write a Python script that collects user data (name, age, email). Add comments explaining how to anonymize or protect this data (e.g., hashing emails, not storing raw values).

Code:



```python
# Collecting user data
name = input("Enter your name: ")
age = input("Enter your age: ")
email = input("Enter your email: ")

# ⚠️ Do not store raw email directly
# Instead, hash sensitive information before storing
import hashlib
hashed_email = hashlib.sha256(email.encode()).hexdigest()

print("User data stored securely (email anonymized).")

```

Output:



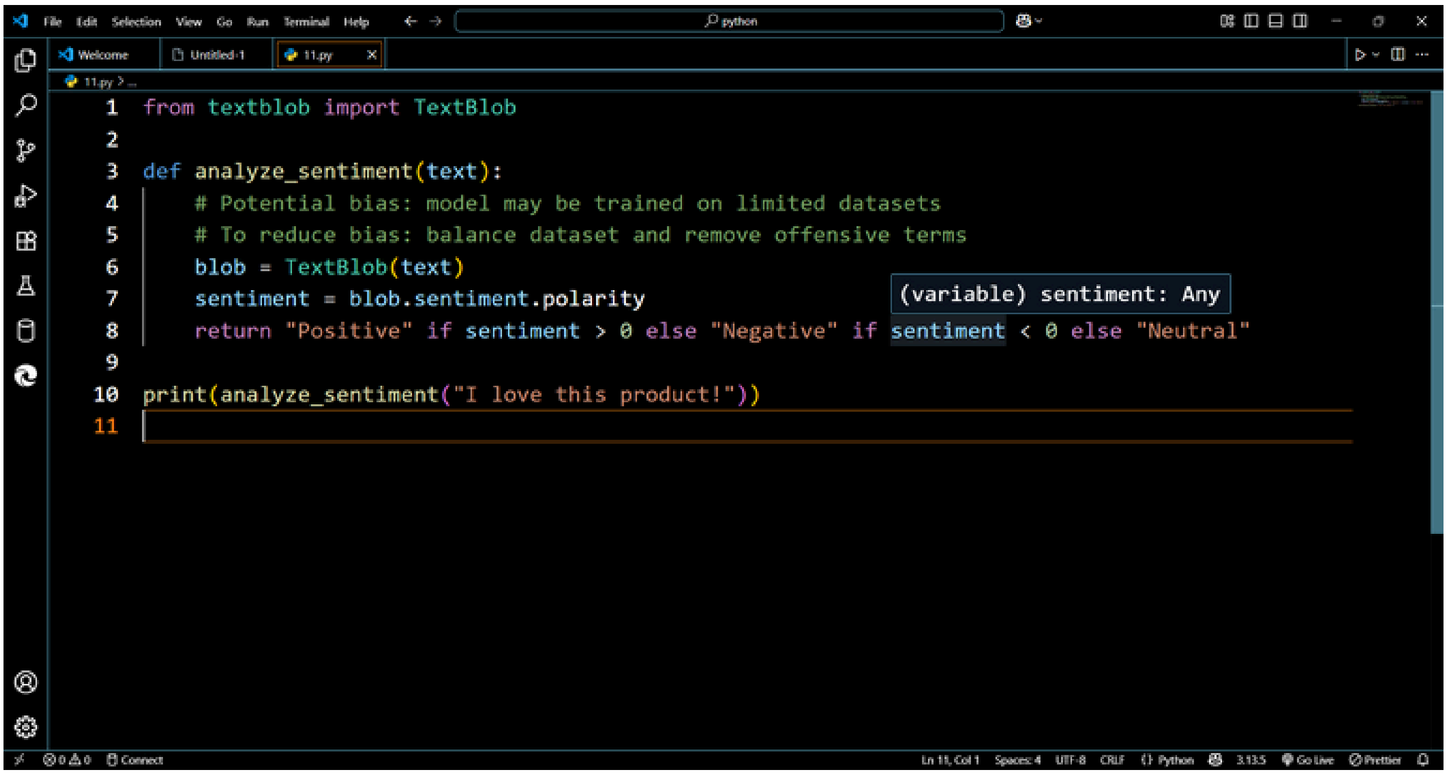
```
PS C:\Users\haasini\OneDrive\Documents\python> python -u "c:\Users\haasini\OneDrive\Documents\python\11.py"
Enter your name: haasini
Enter your age: 18
Enter your email: haasini@gmail.com
User data stored securely (email anonymized).
PS C:\Users\haasini\OneDrive\Documents\python>
```

Explanation: The script collects data but **hashes the email** instead of storing it as plain text. This prevents misuse if data leaks.

TASK:2

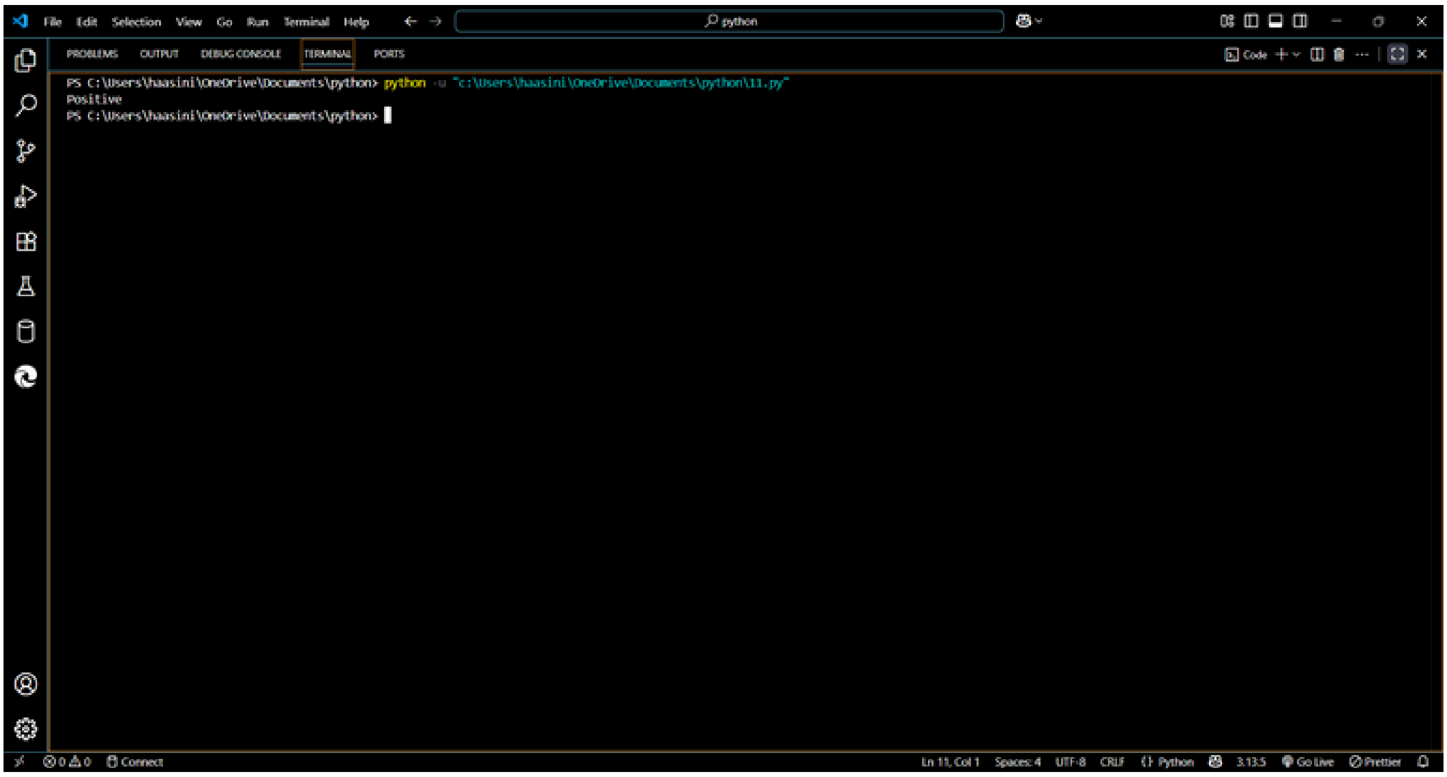
Prompt: Write a Python function for sentiment analysis. Add comments explaining possible biases and strategies like balancing datasets or removing offensive words.

Code:



```python
from textblob import TextBlob

def analyze_sentiment(text):
    # Potential bias: model may be trained on limited datasets
    # To reduce bias: balance dataset and remove offensive terms
    blob = TextBlob(text)
    sentiment = blob.sentiment.polarity
    return "Positive" if sentiment > 0 else "Negative" if sentiment < 0 else "Neutral"

print(analyze_sentiment("I love this product!"))
```

Output:



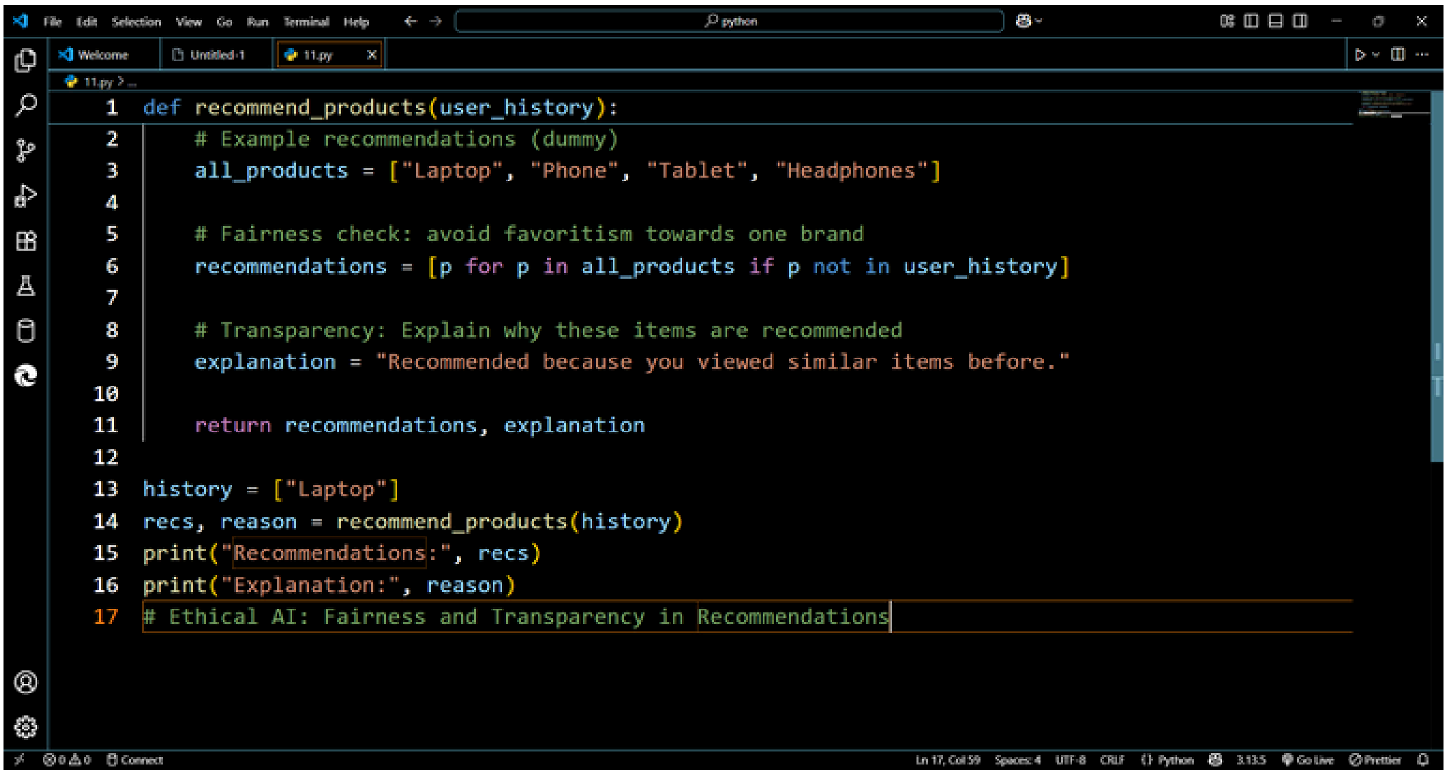
```
PS C:\Users\haasini\OneDrive\Documents\python> python -u "c:\Users\haasini\OneDrive\Documents\python\11.py"
Positive
PS C:\Users\haasini\OneDrive\Documents\python> 
```

Explanation: Uses TextBlob for simple sentiment analysis. Comments highlight **bias risks** and mitigation strategies.

TASK:3

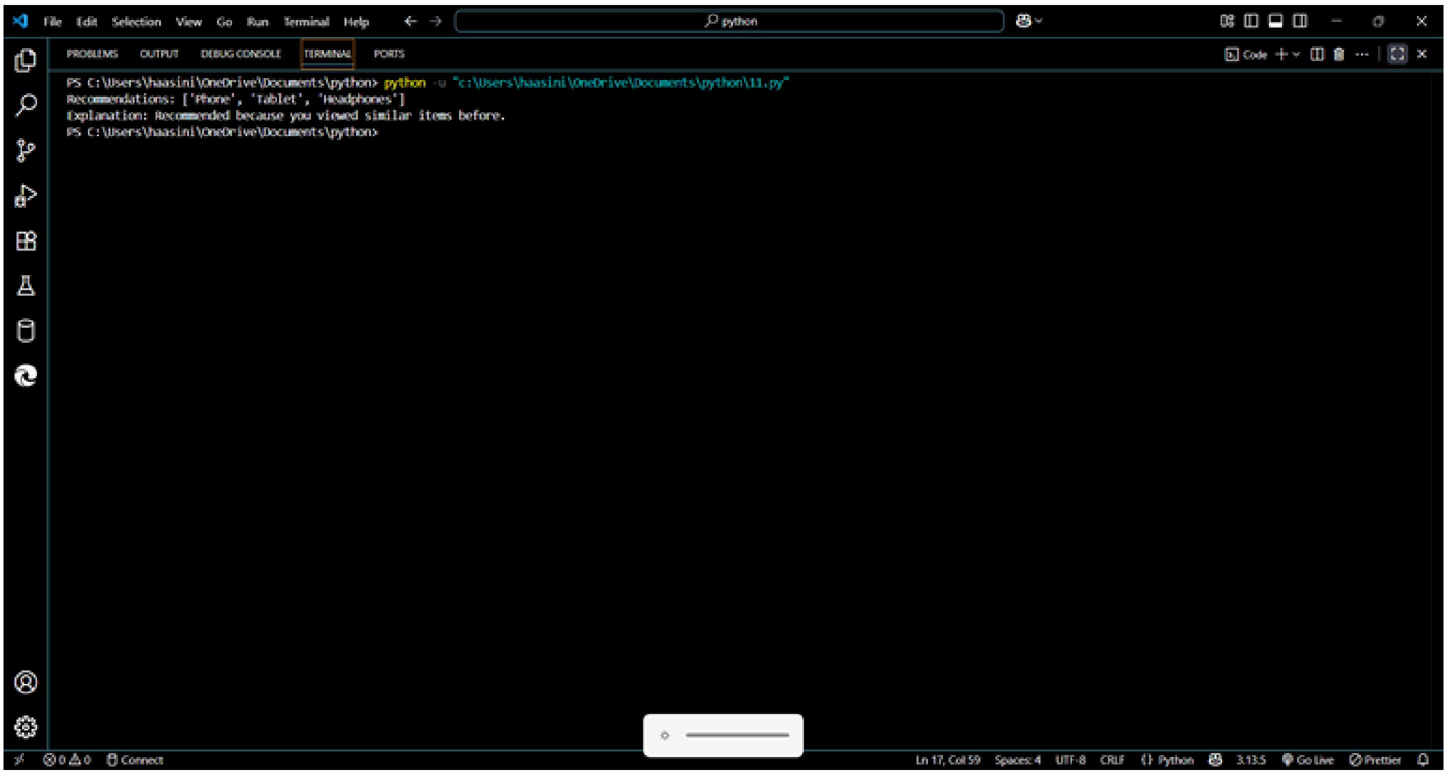
Prompt: Write a Python program that recommends products based on user history. Add fairness checks and user feedback options.

Code:



```python
def recommend_products(user_history):
    # Example recommendations (dummy)
    all_products = ["Laptop", "Phone", "Tablet", "Headphones"]

    # Fairness check: avoid favoritism towards one brand
    recommendations = [p for p in all_products if p not in user_history]

    # Transparency: Explain why these items are recommended
    explanation = "Recommended because you viewed similar items before."

    return recommendations, explanation

history = ["Laptop"]
recs, reason = recommend_products(history)
print("Recommendations:", recs)
print("Explanation:", reason)
# Ethical AI: Fairness and Transparency in Recommendations
```

Output:



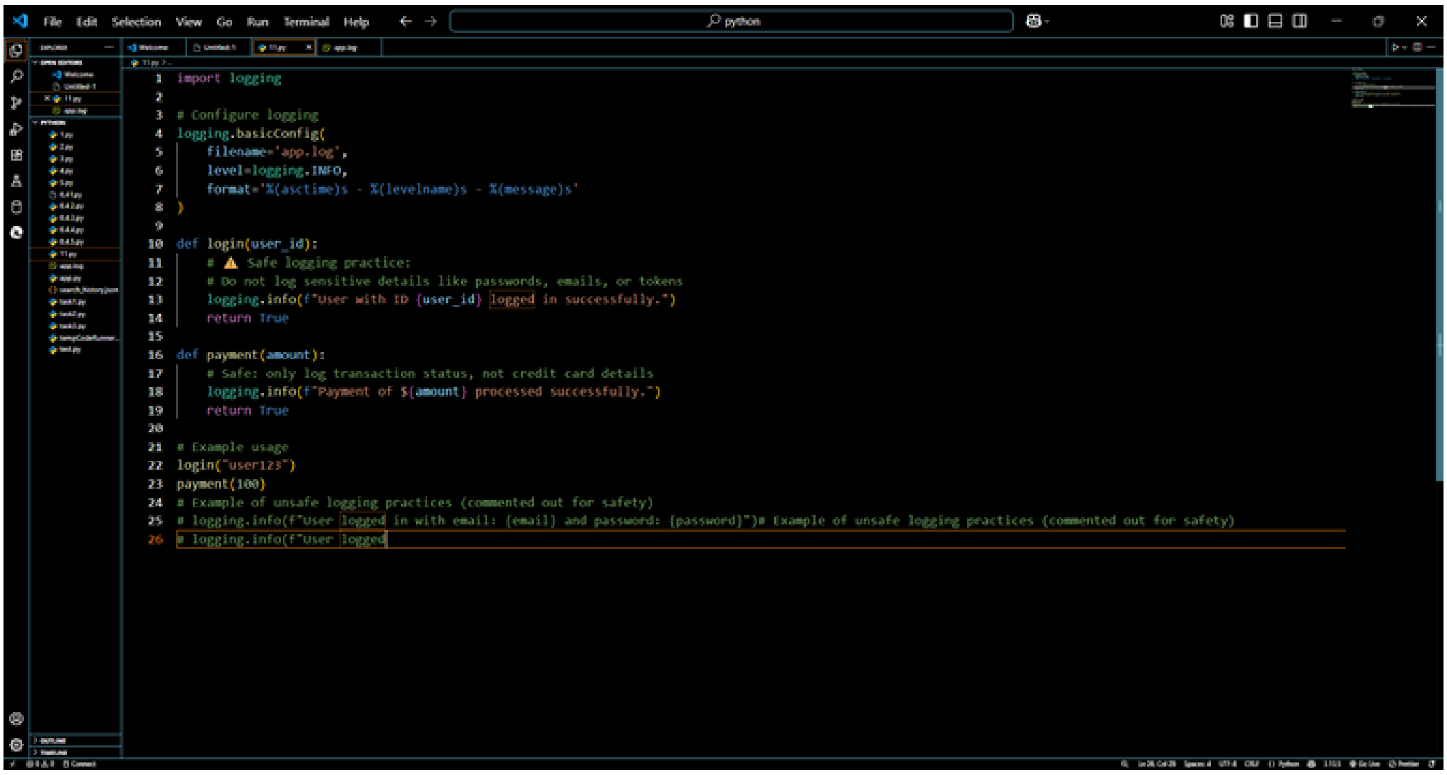
```
PS C:\Users\haasini\OneDrive\Documents\python> python -u "c:\Users\haasini\OneDrive\Documents\python\11.py"
Recommendations: ['Phone', 'Tablet', 'Headphones']
Explanation: Recommended because you viewed similar items before.
PS C:\Users\haasini\OneDrive\Documents\python>
```

Explanation: Ensures **no favoritism** in recommendations and explains **why items are suggested**.

TASK:4

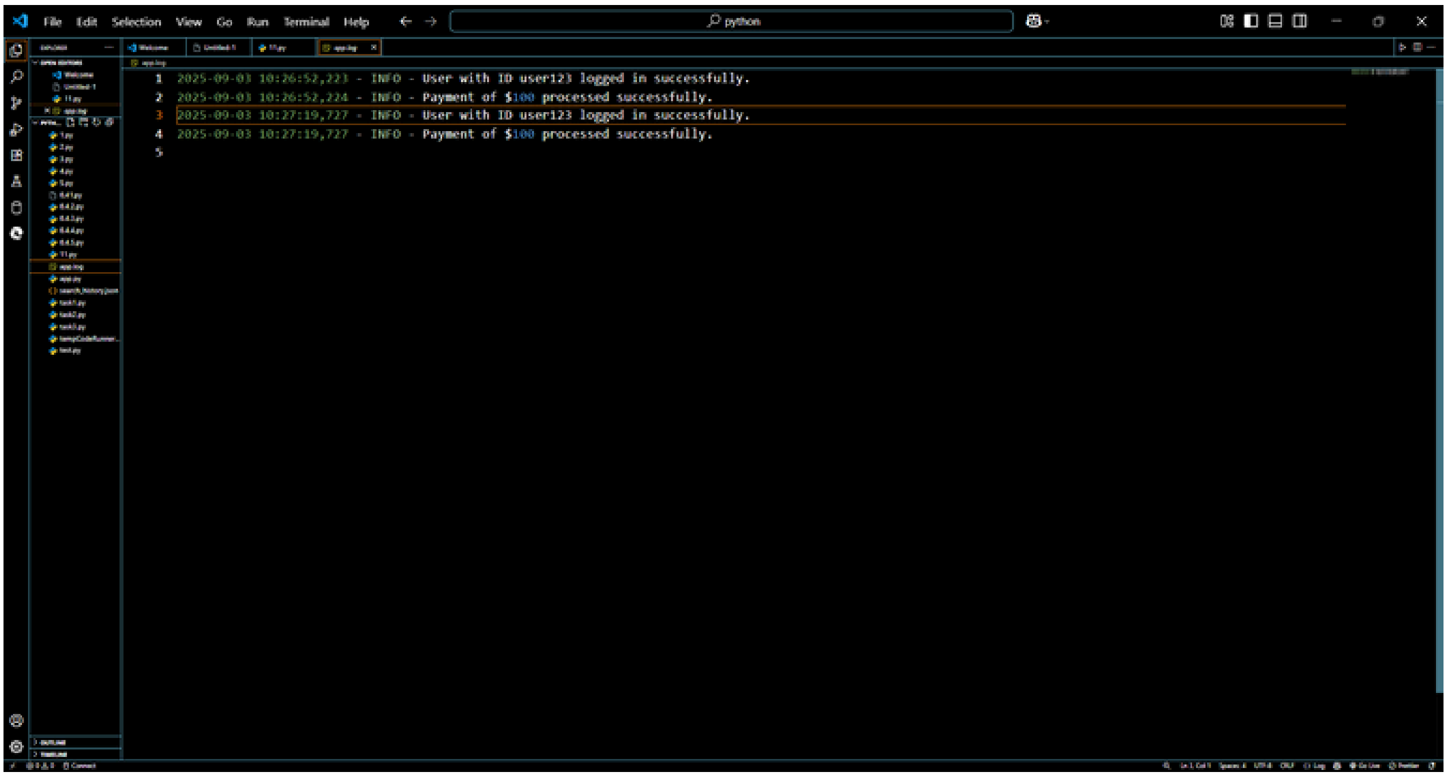
Prompt: Write Python logging code for a web app that avoids storing sensitive information like passwords or emails. Add comments about safe logging practices.

Code:



```python
import logging

# Configure logging
logging.basicConfig(
    filename='app.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

def login(user_id):
    # ⚠️ Safe logging practice:
    # Do not log sensitive details like passwords, emails, or tokens
    logging.info(f"User with ID {user_id} logged in successfully.")
    return True

def payment(amount):
    # Safe: only log transaction status, not credit card details
    logging.info(f"Payment of ${amount} processed successfully.")
    return True

# Example usage
login("user123")
payment(100)
# Example of unsafe logging practices (commented out for safety)
# logging.info(f"User logged in with email: {email} and password: {password}")# Example of unsafe logging practices (commented out for safety)
# logging.info(f"User logged
```

Output:



```
2025-09-03 10:26:52,223 - INFO - User with ID user123 logged in successfully.
2025-09-03 10:26:52,224 - INFO - Payment of $100 processed successfully.
2025-09-03 10:27:19,727 - INFO - User with ID user123 logged in successfully.
2025-09-03 10:27:19,727 - INFO - Payment of $100 processed successfully.
```

Explanation:

We use **logging module** to record events (like logins, payments).

Sensitive data (passwords, emails, credit card info) is **not logged**.

Logs are stored in a file (app.log) instead of printing to terminal (can be changed).

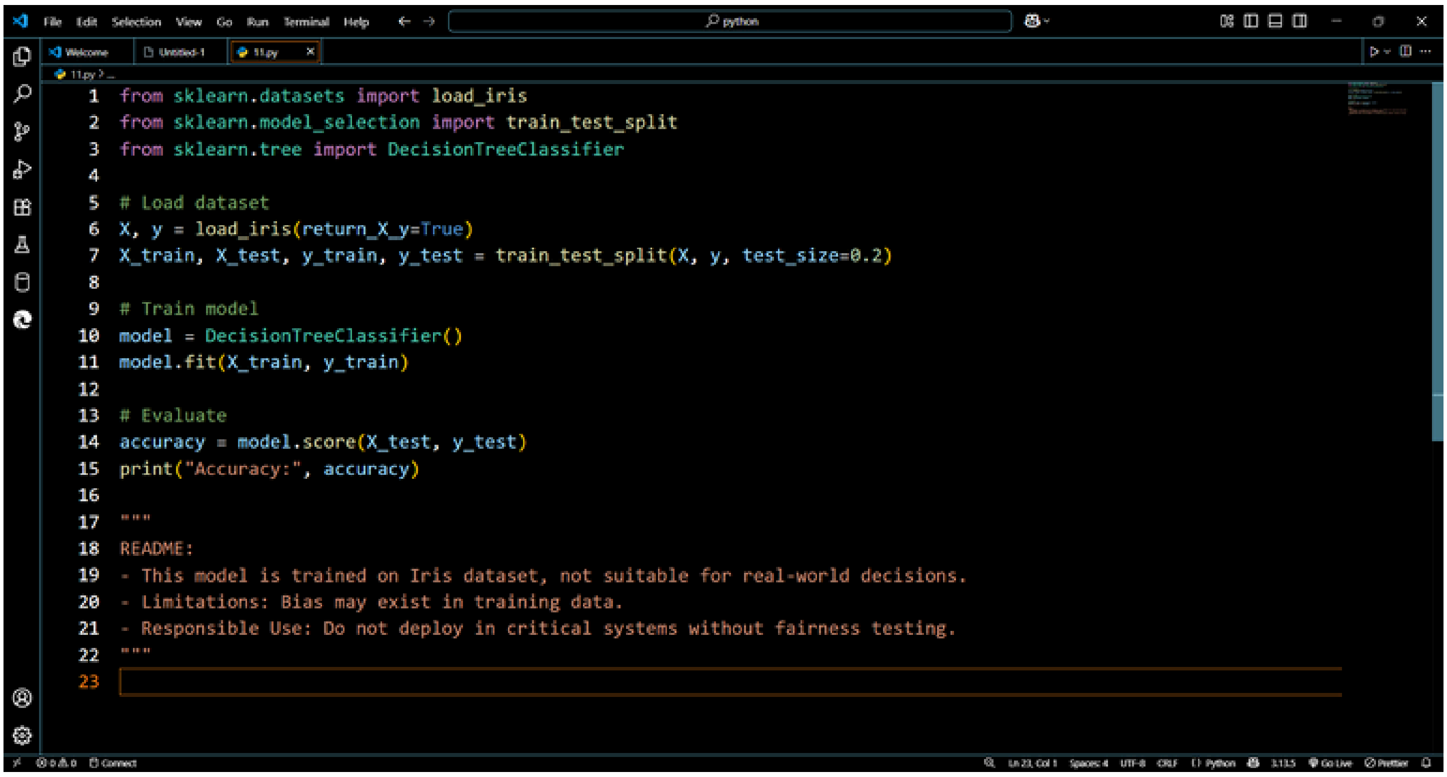
Example:

- logging.info("User logged in") → records safe info only.
- logging.info("Payment processed") → records transaction status, not details.

TASK:5

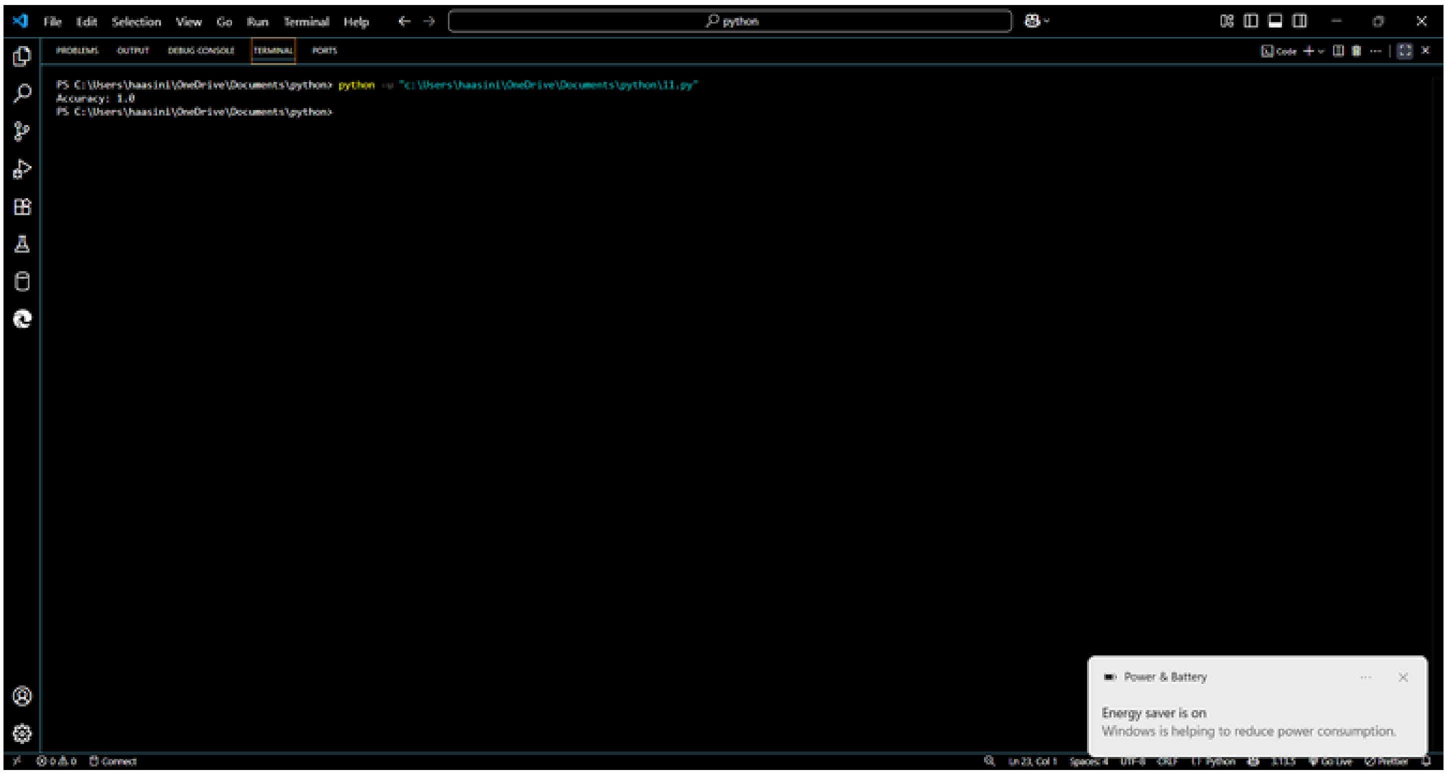
Prompt: Write Python code to train a simple ML model on sample data. Add a README-style comment about responsible usage, fairness, and limitations.

Code:



```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# Load dataset
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Evaluate
accuracy = model.score(X_test, y_test)
print("Accuracy:", accuracy)

"""
README:
- This model is trained on Iris dataset, not suitable for real-world decisions.
- Limitations: Bias may exist in training data.
- Responsible Use: Do not deploy in critical systems without fairness testing.
"""
```

Output:



```
PS C:\Users\haasini\OneDrive\Documents\python> python -u "c:\Users\haasini\OneDrive\Documents\python\11.py"
Accuracy: 1.0
PS C:\Users\haasini\OneDrive\Documents\python>
```

Power & Battery

Energy saver is on
Windows is helping to reduce power consumption.

Explanation:

Trains a simple ML model and includes a **README-style note** about responsible AI use.