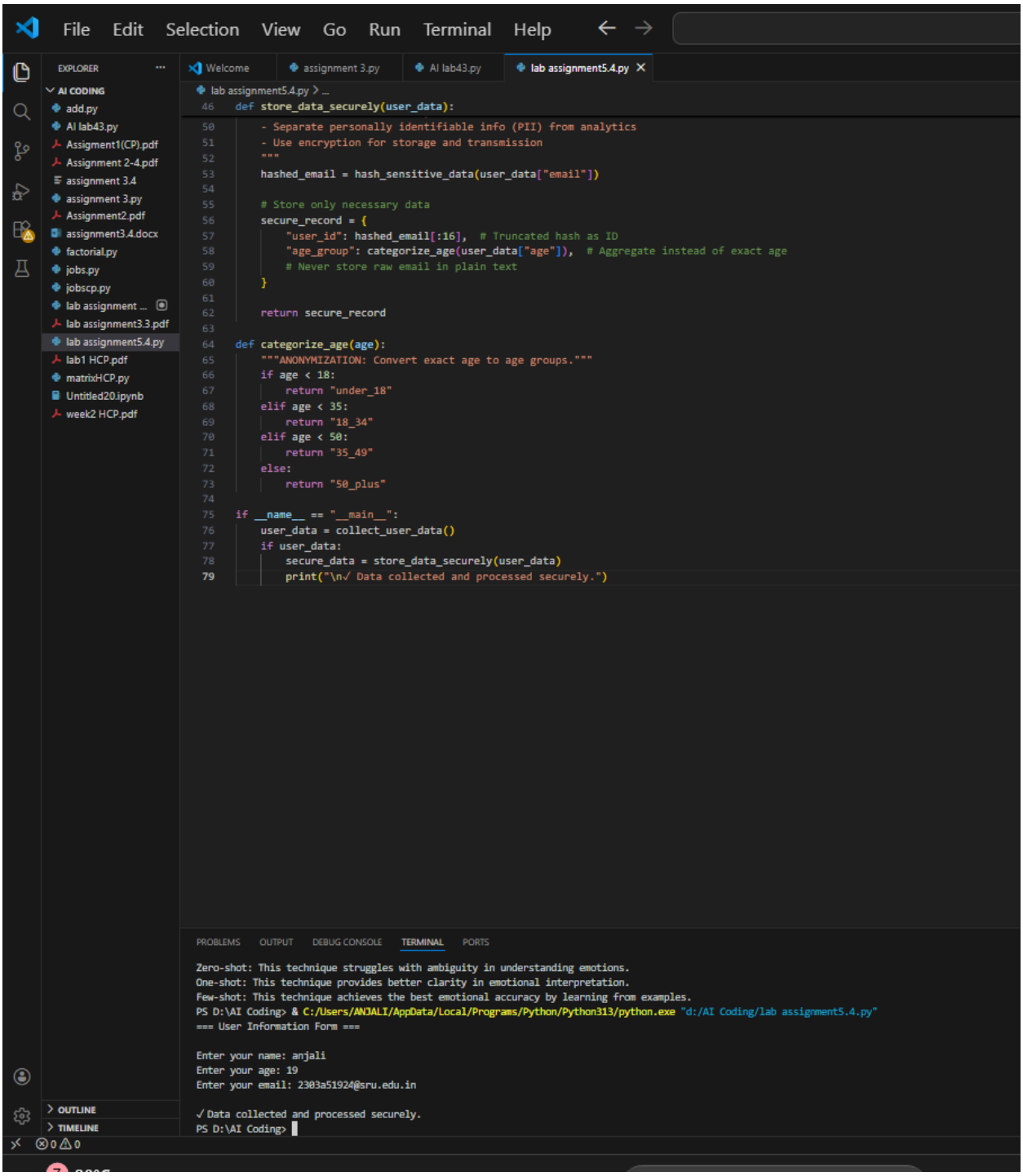
2303a51504

Lab assignment 5.4

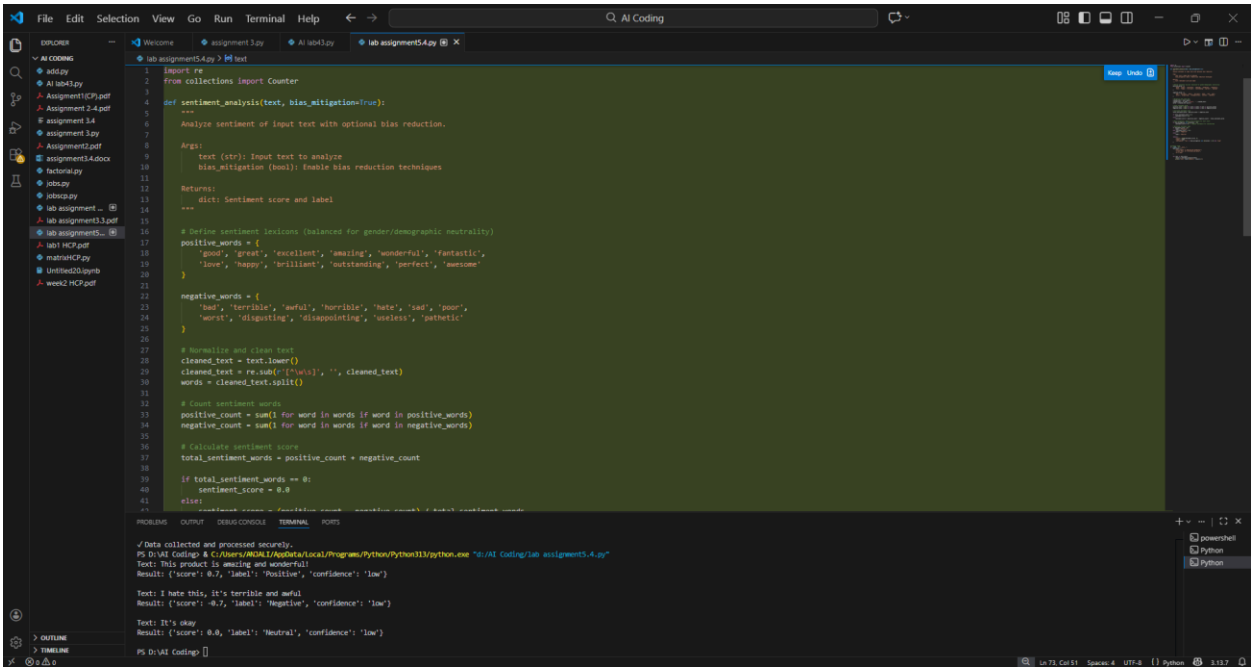
Lab 5: Ethical Foundations – Responsible AI Coding Practices

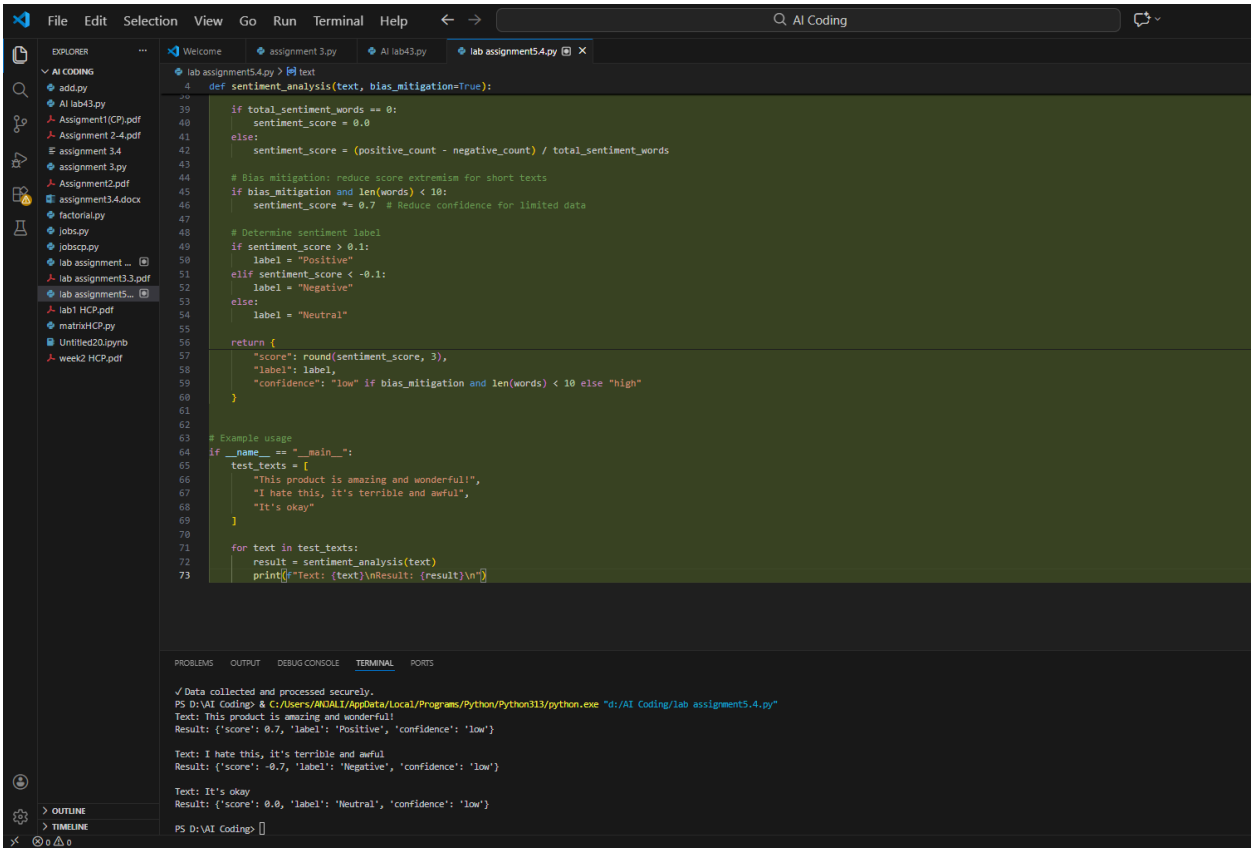
1)Prompt GitHub Copilot to generate a Python script that collects user data (e.g., name, age, email). Then, ask Copilot to add comments on how to anonymize or protect this data.



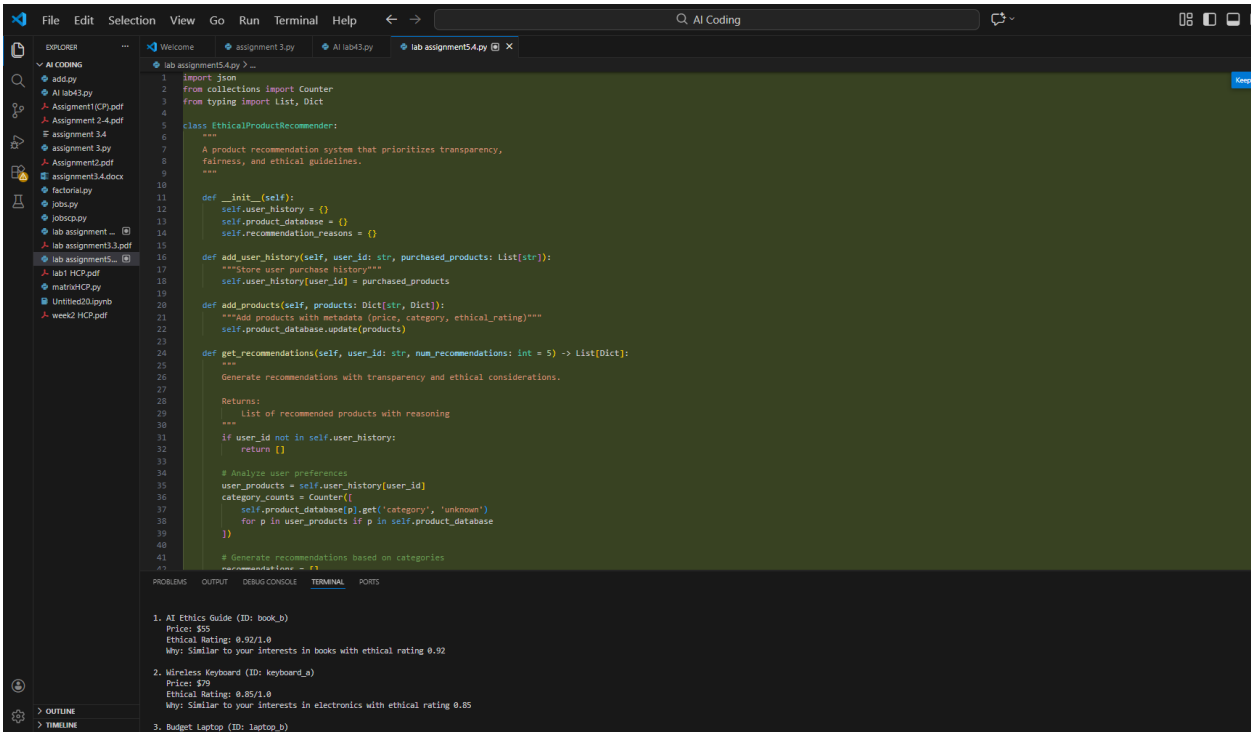
2) Task Description #2:

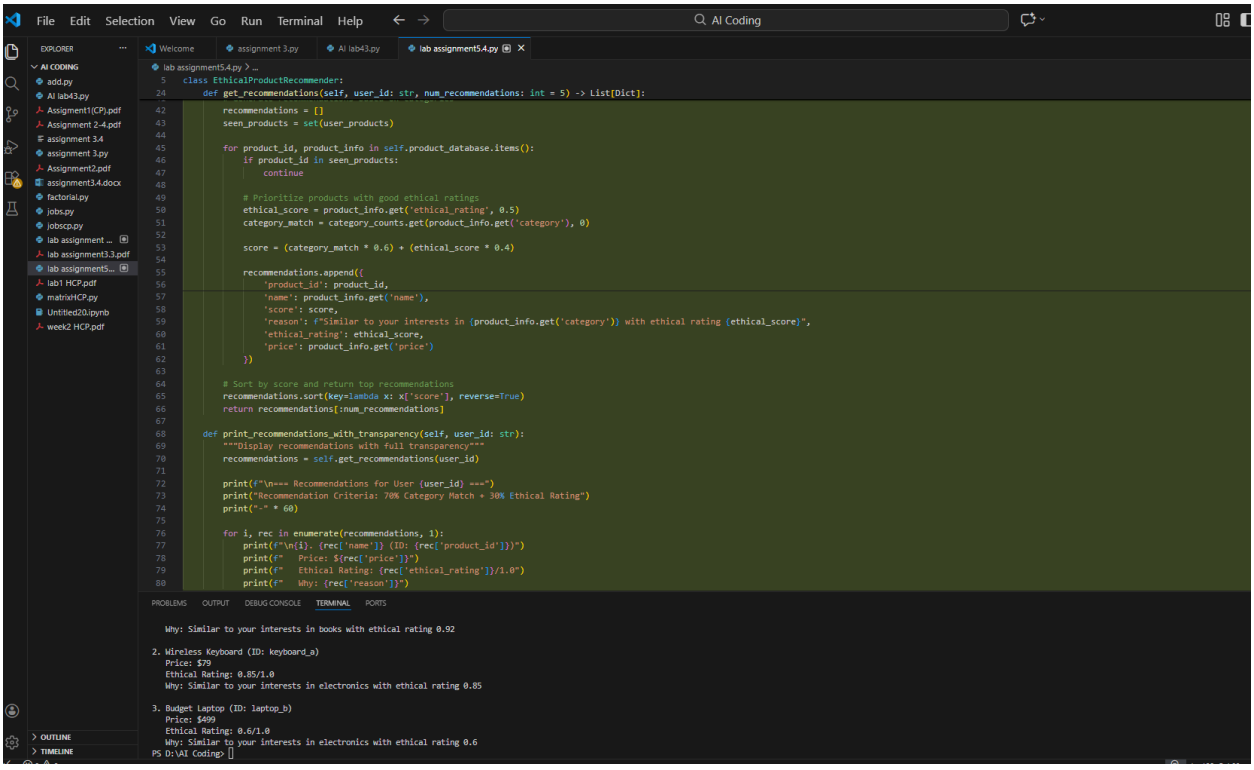
• Ask Copilot to generate a Python function for sentiment analysis. Then prompt Copilot to identify and handle potential biases in the data.

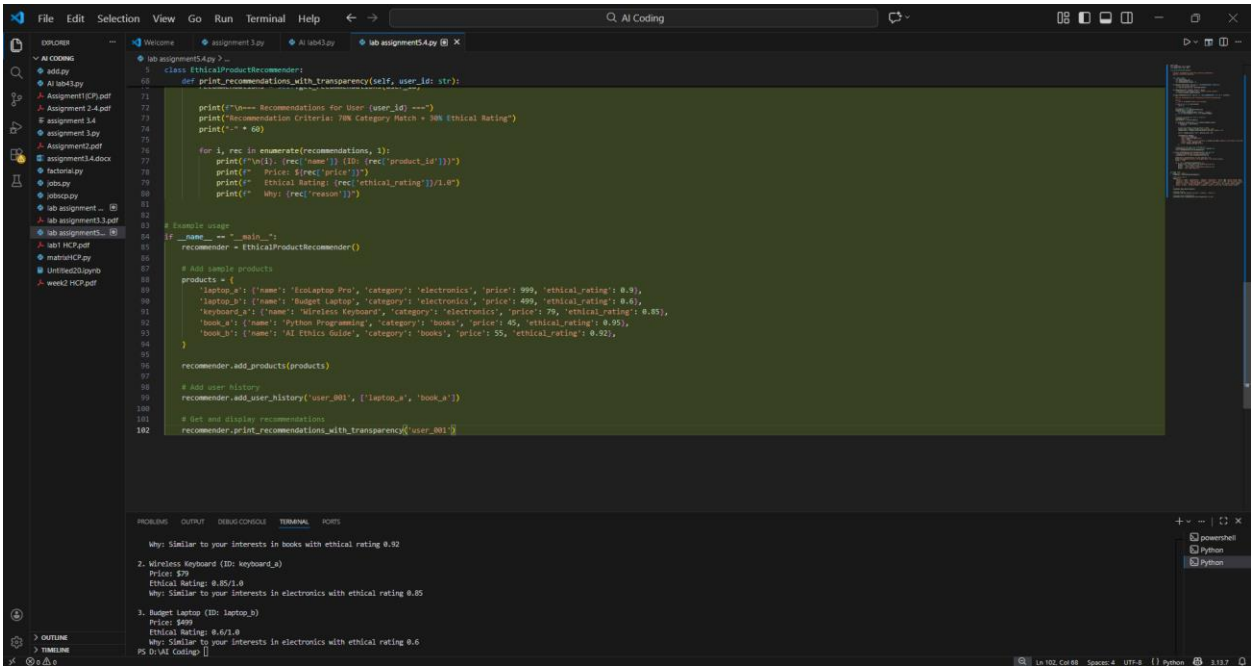




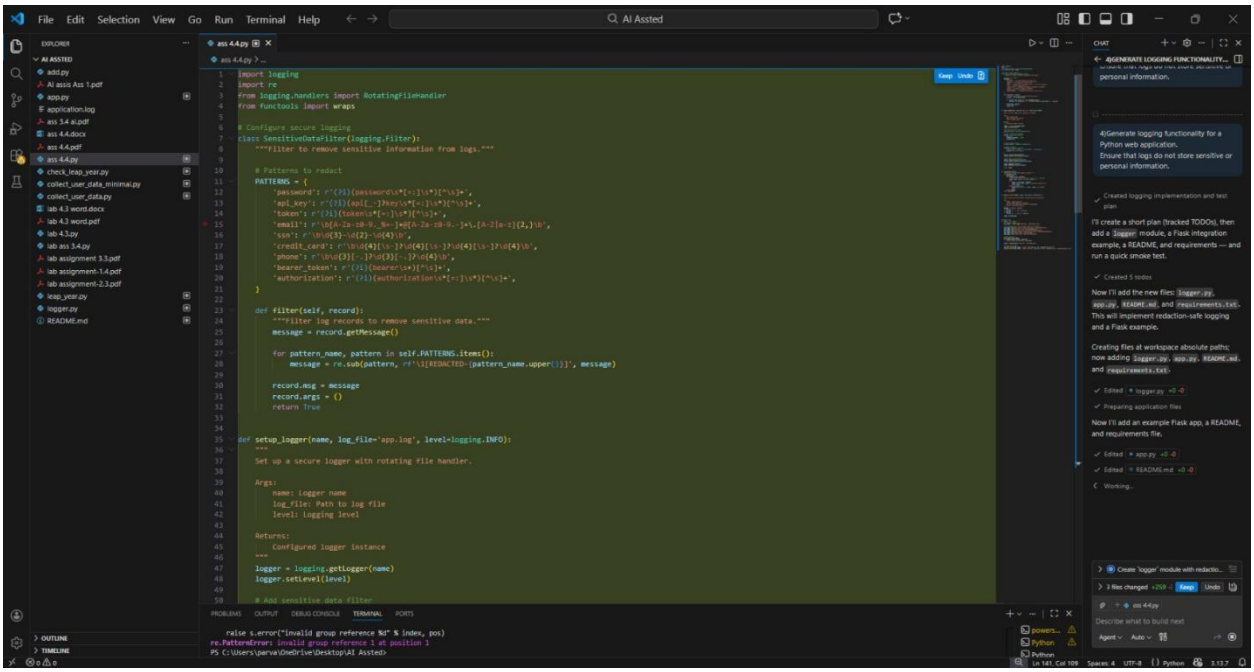
3) Use Copilot to write a Python program that recommends products based on user history. Ask it to follow ethical guidelines like transparency and fairness.

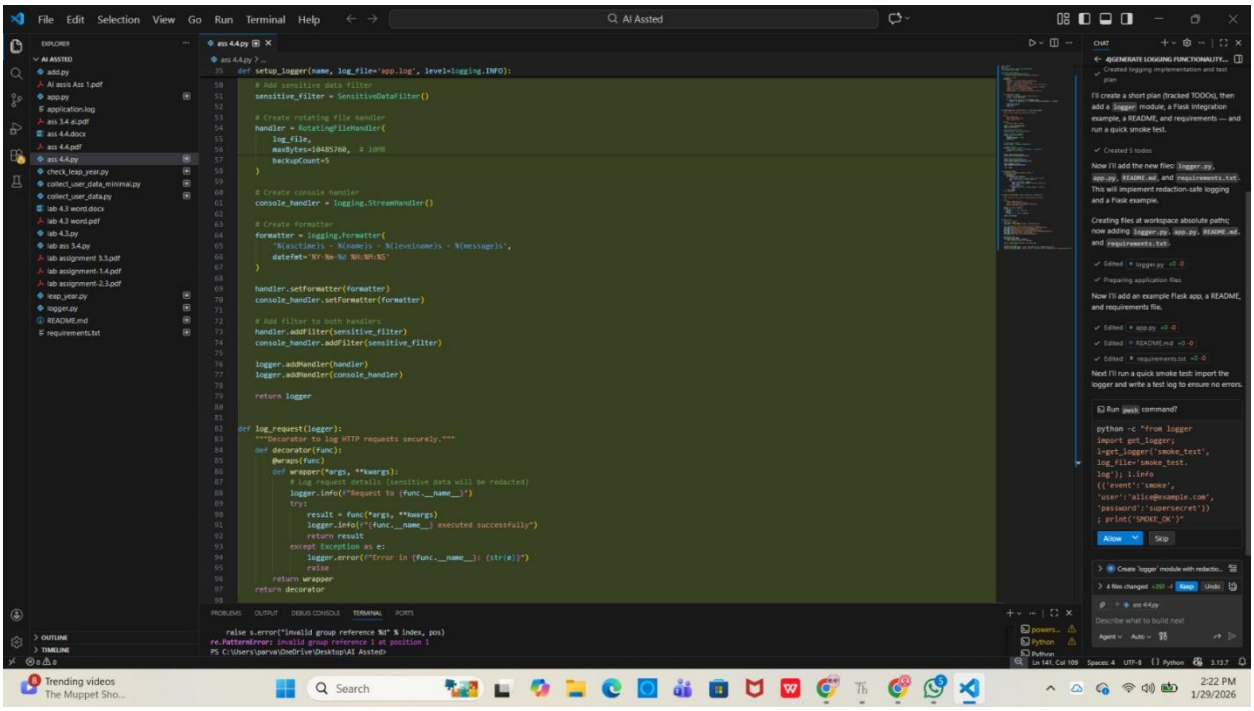

File Edit Selection View Go Run Terminal Help
AI Coding
EXPLORER
AI CODING
add.py
AI lab43.py
Assigment1(CP).pdf
Assignment 2-4.pdf
assignment 3.4
assignment 3.py
Assignment2.pdf
assignment3.4.docx
factorial.py
jobs.py
jobscp.py
lab assignment ...
lab assignment3.3.pdf
lab assignment5...
lab1 HCP.pdf
matrixHCP.py
Untitled20.ipynb
week2 HCP.pdf
Welcome
assignment 3.py
AI lab43.py
lab assignment5.4.py
lab assignment5.4.py > ...
import json
from collections import Counter
from typing import List, Dict
class EthicalProductRecommender:
"""
A product recommendation system that prioritizes transparency,
fairness, and ethical guidelines.
"""
def __init__(self):
self.user_history = {}
self.product_database = {}
self.recommendation_reasons = {}
def add_user_history(self, user_id: str, purchased_products: List[str]):
"""Store user purchase history"""
self.user_history[user_id] = purchased_products
def add_products(self, products: Dict[str, Dict]):
"""Add products with metadata (price, category, ethical_rating)"""
self.product_database.update(products)
def get_recommendations(self, user_id: str, num_recommendations: int = 5) -> List[Dict]:
"""
Generate recommendations with transparency and ethical considerations.
Returns:
List of recommended products with reasoning
"""
if user_id not in self.user_history:
return []
# Analyze user preferences
user_products = self.user_history[user_id]
category_counts = Counter([
self.product_database[p].get('category', 'unknown')
for p in user_products if p in self.product_database
])
Generate recommendations based on categories
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
1. AI Ethics Guide (ID: book_b)
Price: $55
Ethical Rating: 0.92/1.0
Why: Similar to your interests in books with ethical rating 0.92
2. Wireless Keyboard (ID: keyboard_a)
Price: $79
Ethical Rating: 0.85/1.0
Why: Similar to your interests in electronics with ethical rating 0.85
3. Budget Laptop (ID: laptop_b)
OUTLINE
TIMELINE


File Edit Selection View Go Run Terminal Help
AI Coding
EXPLORER
AI CODING
add.py
AI lab43.py
Assigment1(CP).pdf
Assignment 2-4.pdf
assignment 3.4
assignment 3.py
Assignment2.pdf
assignment3.4.docx
factorial.py
jobs.py
jobscp.py
lab assignment ...
lab assignment3.3.pdf
lab assignment5...
lab1 HCP.pdf
matrixHCP.py
Untitled20.ipynb
week2 HCP.pdf
Welcome
assignment 3.py
AI lab43.py
lab assignment5.4.py
lab assignment5.4.py > ...
class EthicalProductRecommender:
def get_recommendations(self, user_id: str, num_recommendations: int = 5) -> List[Dict]:
recommendations = []
seen_products = set(user_products)
for product_id, product_info in self.product_database.items():
if product_id in seen_products:
continue
# Prioritize products with good ethical ratings
ethical_score = product_info.get('ethical_rating', 0.5)
category_match = category_counts.get(product_info.get('category'), 0)
score = (category_match * 0.6) + (ethical_score * 0.4)
recommendations.append({
'product_id': product_id,
'name': product_info.get('name'),
'score': score,
'reason': f"Similar to your interests in {product_info.get('category')} with ethical rating {ethical_score}",
'ethical_rating': ethical_score,
'price': product_info.get('price')
})
# Sort by score and return top recommendations
recommendations.sort(key=lambda x: x['score'], reverse=True)
return recommendations[:num_recommendations]
def print_recommendations_with_transparency(self, user_id: str):
"""Display recommendations with full transparency"""
recommendations = self.get_recommendations(user_id)
print(f"\n=== Recommendations for User {user_id} ===")
print("Recommendation Criteria: 70% Category Match + 30% Ethical Rating")
print("-" * 60)
for i, rec in enumerate(recommendations, 1):
print(f"\n{i}. {rec['name']} (ID: {rec['product_id']})")
print(f" Price: ${rec['price']}")
print(f" Ethical Rating: {rec['ethical_rating']}/1.0")
print(f" Why: {rec['reason']}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Why: Similar to your interests in books with ethical rating 0.92
2. Wireless Keyboard (ID: keyboard_a)
Price: $79
Ethical Rating: 0.85/1.0
Why: Similar to your interests in electronics with ethical rating 0.85
3. Budget Laptop (ID: laptop_b)
Price: $499
Ethical Rating: 0.6/1.0
Why: Similar to your interests in electronics with ethical rating 0.6
PS D:\AI Coding>
OUTLINE
TIMELINE

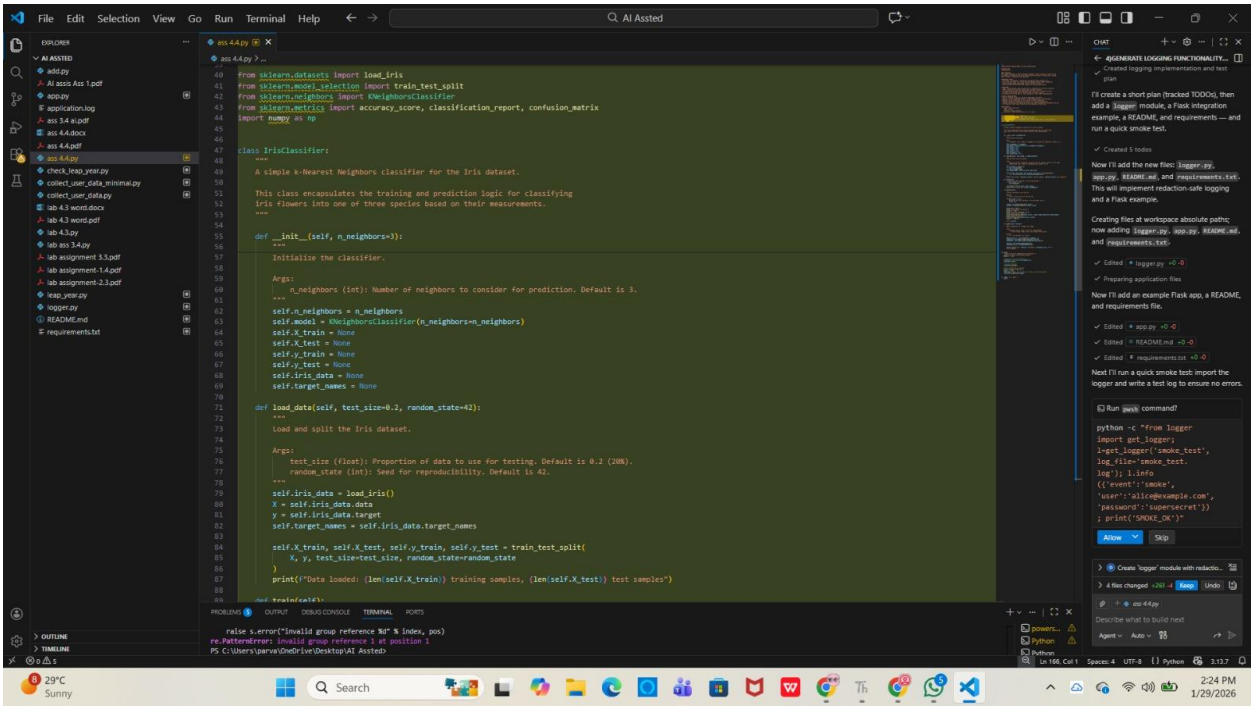


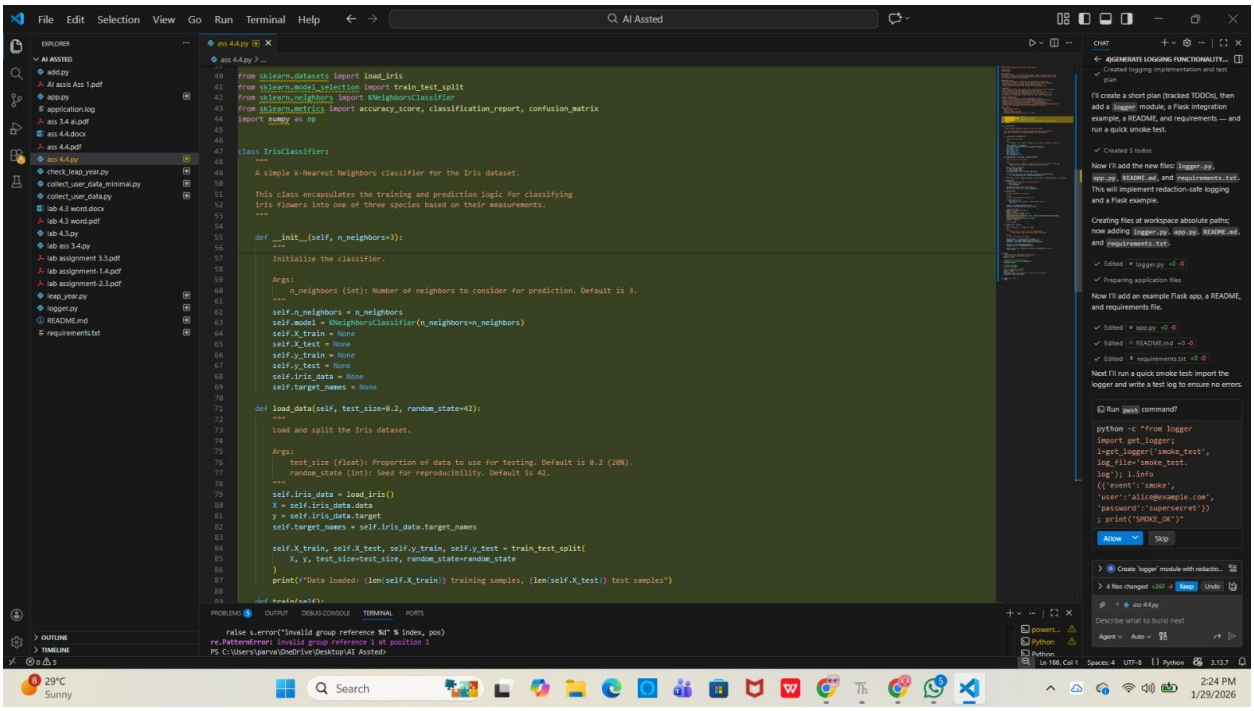
4) Prompt Copilot to generate logging functionality in a Python web application. Then, ask it to ensure the logs do not record sensitive information.

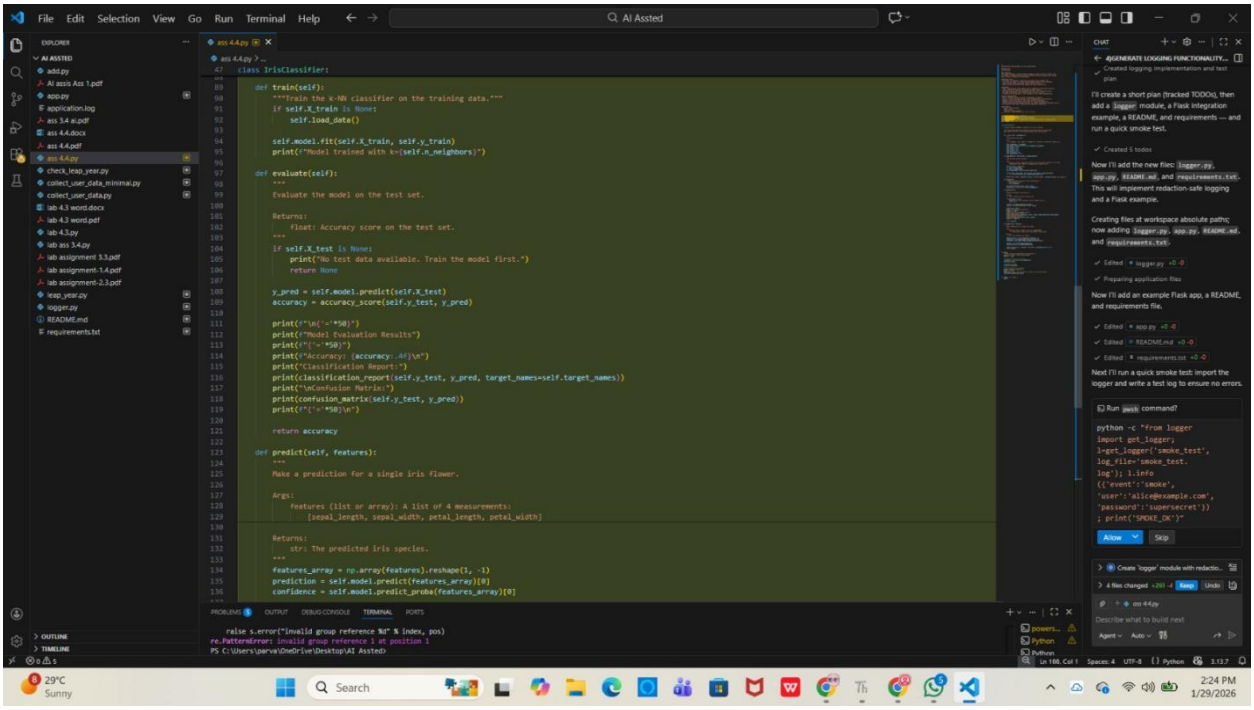




5)Ask Copilot to generate a machine learning model. Then, prompt it to add documentation on how to use the model responsibly (e.g., explainability, accuracy limits).




File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import numpy as np
class IrisClassifier:
"""
A simple k-Nearest Neighbors classifier for the Iris dataset.
This class encapsulates the training and prediction logic for classifying
iris flowers into one of three species based on their measurements.
"""
def __init__(self, n_neighbors=3):
"""
Initialize the classifier.
Args:
n_neighbors (int): Number of neighbors to consider for prediction. Default is 3.
"""
self.n_neighbors = n_neighbors
self.model = KNeighborsClassifier(n_neighbors=n_neighbors)
self.X_train = None
self.X_test = None
self.y_train = None
self.y_test = None
self.iris_data = None
self.target_names = None
def load_data(self, test_size=0.2, random_state=42):
"""
Load and split the Iris dataset.
Args:
test_size (float): Proportion of data to use for testing. Default is 0.2 (20%).
random_state (int): Seed for reproducibility. Default is 42.
"""
self.iris_data = load_iris()
X = self.iris_data.data
y = self.iris_data.target
self.target_names = self.iris_data.target_names
self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(
X, y, test_size=test_size, random_state=random_state
)
print(f"Data loaded: {len(self.X_train)} training samples, {len(self.X_test)} test samples")
def train(self):
raise s.error("invalid group reference %d" % index, pos)
re.PatternError: invalid group reference 1 at position 1
PS C:\Users\parva\OneDrive\Desktop\AI Assted>
python -c "from logger import get_logger; l=get_logger('smoke_test', log_file='smoke_test.log'); l.info({'event':'smoke', 'user':'alice@example.com', 'password':'supersecret'}); print('SMOKE_OK')"
Allow
Skip
29°C
Sunny
Search
2:24 PM
1/29/2026


File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
class IrisClassifier:
def train(self):
"""Train the k-NN classifier on the training data."""
if self.X_train is None:
self.load_data()
self.model.fit(self.X_train, self.y_train)
print(f"Model trained with k={self.n_neighbors}")
def evaluate(self):
"""
Evaluate the model on the test set.
Returns:
float: Accuracy score on the test set.
"""
if self.X_test is None:
print("No test data available. Train the model first.")
return None
y_pred = self.model.predict(self.X_test)
accuracy = accuracy_score(self.y_test, y_pred)
print(f"\n{'='*50}")
print(f"Model Evaluation Results")
print(f"{'='*50}")
print(f"Accuracy: {accuracy:.4f}\n")
print("Classification Report:")
print(classification_report(self.y_test, y_pred, target_names=self.target_names))
print("\nConfusion Matrix:")
print(confusion_matrix(self.y_test, y_pred))
print(f"{'='*50}\n")
return accuracy
def predict(self, features):
"""
Make a prediction for a single iris flower.
Args:
features (list or array): A list of 4 measurements:
[sepal_length, sepal_width, petal_length, petal_width]
Returns:
str: The predicted iris species.
"""
features_array = np.array(features).reshape(1, -1)
prediction = self.model.predict(features_array)[0]
confidence = self.model.predict_proba(features_array)[0]
raise s.error("invalid group reference %d" % index, pos)
re.PatternError: invalid group reference 1 at position 1
PS C:\Users\parva\OneDrive\Desktop\AI Assted>
python -c "from logger import get_logger; l=get_logger('smoke_test', log_file='smoke_test.log'); l.info({'event':'smoke', 'user':'alice@example.com', 'password':'supersecret'}); print('SMOKE_OK')"
Allow
Skip
29°C
Sunny
Search
2:24 PM
1/29/2026