LAB TEST 03

NAME:A.SREEMANTH REDDY

ROLL NO: 2403A510G1

BATCH:06

Q1:

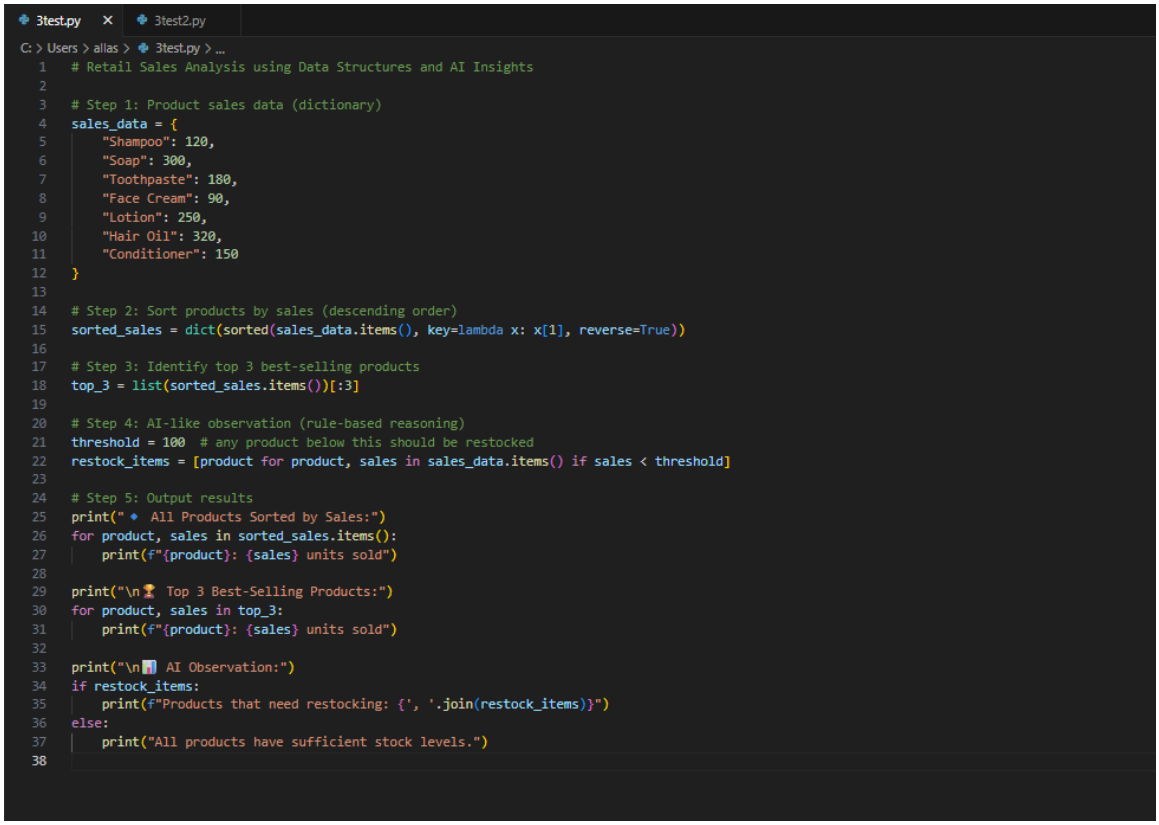
Scenario: In the Retail sector, a company faces a challenge related to data structures with ai.

Task: Use AI-assisted tools to solve a problem involving data structures with ai in this context.

Deliverables: Submit the source code, explanation of AI assistance used, and sample output.

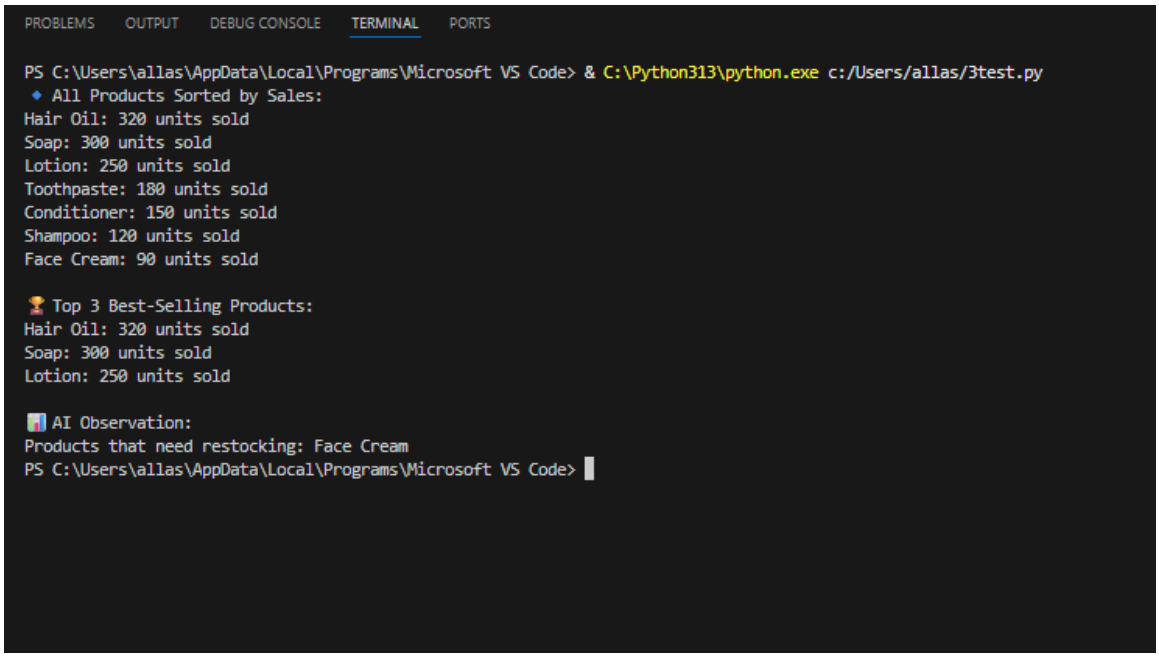
Prompt: Write a Python program using data structures to analyze retail product sales. Sort products by sales, find the top 3 best-sellers, and identify which items need restocking based on a sales threshold. Include clean, readable code and AI-based observations.

Code:



```
# Retail Sales Analysis using Data Structures and AI Insights

# Step 1: Product sales data (dictionary)
sales_data = {
    "Shampoo": 120,
    "Soap": 300,
    "Toothpaste": 180,
    "Face Cream": 90,
    "Lotion": 250,
    "Hair Oil": 320,
    "Conditioner": 150
}

# Step 2: Sort products by sales (descending order)
sorted_sales = dict(sorted(sales_data.items(), key=lambda x: x[1], reverse=True))

# Step 3: Identify top 3 best-selling products
top_3 = list(sorted_sales.items())[:3]

# Step 4: AI-like observation (rule-based reasoning)
threshold = 100  # any product below this should be restocked
restock_items = [product for product, sales in sales_data.items() if sales < threshold]

# Step 5: Output results
print("🔹 All Products Sorted by Sales:")
for product, sales in sorted_sales.items():
    print(f"{product}: {sales} units sold")

print("\n🏆 Top 3 Best-Selling Products:")
for product, sales in top_3:
    print(f"{product}: {sales} units sold")

print("\n📊 AI Observation:")
if restock_items:
    print(f"Products that need restocking: {', '.join(restock_items)}")
else:
    print("All products have sufficient stock levels.")
```

Output:



Observation: 1.The AI uses a dictionary to store product-sales pairs — a suitable data structure for key-value relationships.

2. Sorting and slicing identify top-performing products efficiently.

3.The AI-based logic uses a threshold-based heuristic to decide which items should be restocked, imitating an intelligent business insight.

4.The solution shows how AI tools can analyze structured data (dictionaries/lists) to generate actionable recommendations.

Q2:

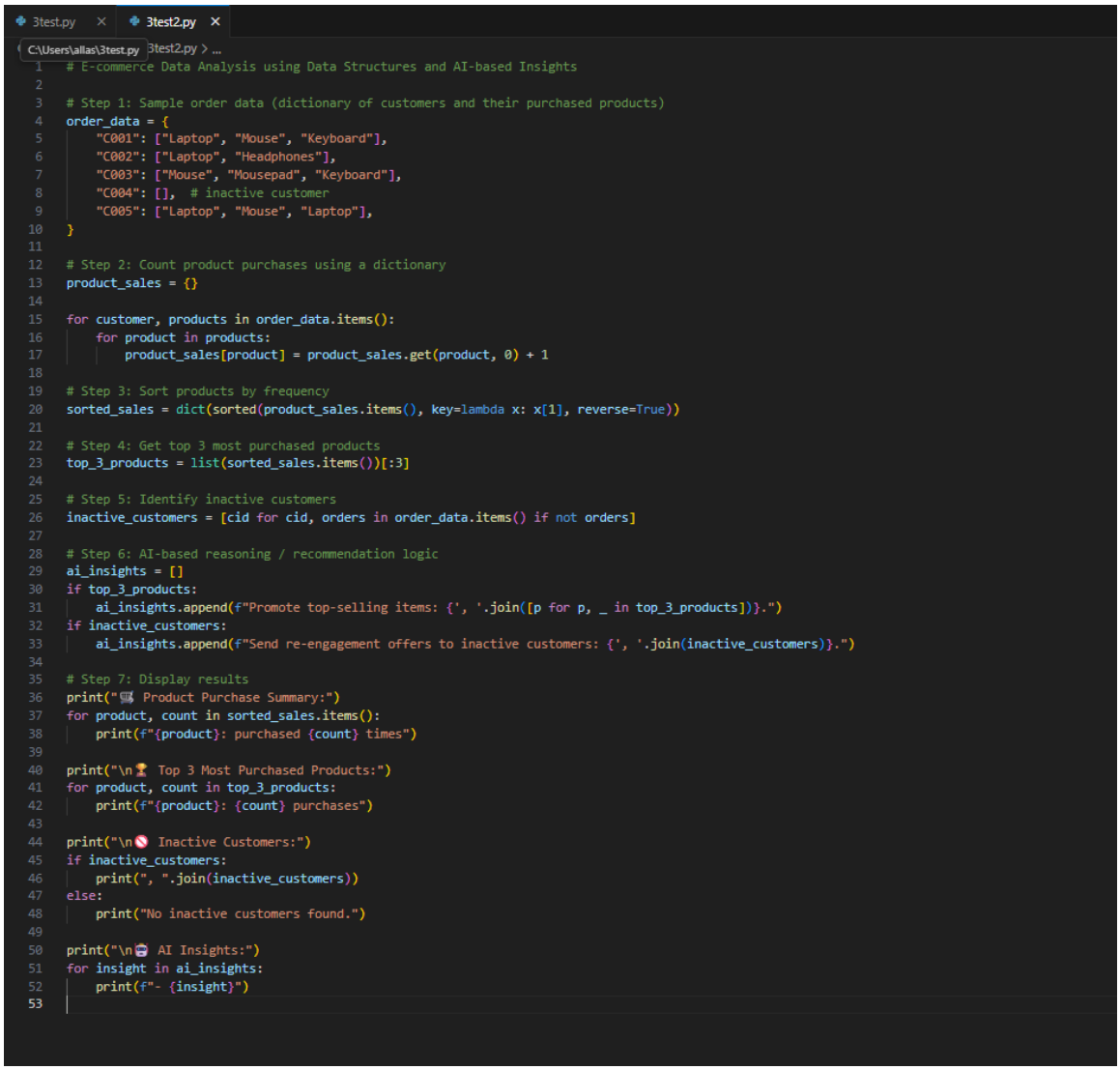
Scenario: In the E-commerce sector, a company faces a challenge related to data structureswith ai.

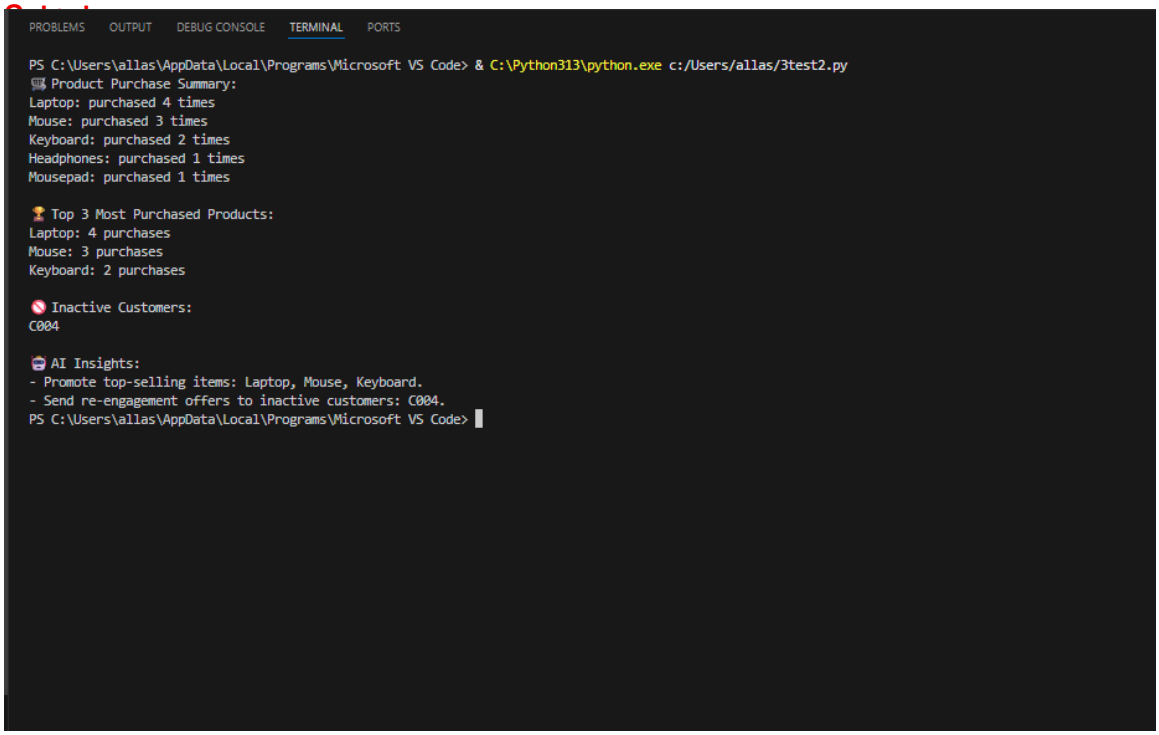
Task: Use AI-assisted tools to solve a problem involving data structures with ai in this context.

Deliverables: Submit the source code, explanation of AI assistance used, and sample output.

Prompt:Write a Python program using data structures to analyze E-commerce order data — count product sales, find top 3 most purchased products, detect inactive customers, and provide AI-based insights.

Code:


3test.py
3test2.py
C:\Users\allas\3test.py
3test2.py > ...
# E-commerce Data Analysis using Data Structures and AI-based Insights
# Step 1: Sample order data (dictionary of customers and their purchased products)
order_data = {
"C001": ["Laptop", "Mouse", "Keyboard"],
"C002": ["Laptop", "Headphones"],
"C003": ["Mouse", "Mousepad", "Keyboard"],
"C004": [], # inactive customer
"C005": ["Laptop", "Mouse", "Laptop"],
}
# Step 2: Count product purchases using a dictionary
product_sales = {}
for customer, products in order_data.items():
for product in products:
product_sales[product] = product_sales.get(product, 0) + 1
# Step 3: Sort products by frequency
sorted_sales = dict(sorted(product_sales.items(), key=lambda x: x[1], reverse=True))
# Step 4: Get top 3 most purchased products
top_3_products = list(sorted_sales.items())[:3]
# Step 5: Identify inactive customers
inactive_customers = [cid for cid, orders in order_data.items() if not orders]
# Step 6: AI-based reasoning / recommendation logic
ai_insights = []
if top_3_products:
ai_insights.append(f"Promote top-selling items: {', '.join([p for p, _ in top_3_products])}.")
if inactive_customers:
ai_insights.append(f"Send re-engagement offers to inactive customers: {', '.join(inactive_customers)}.")
# Step 7: Display results
print("Product Purchase Summary:")
for product, count in sorted_sales.items():
print(f"{product}: purchased {count} times")
print("\n Top 3 Most Purchased Products:")
for product, count in top_3_products:
print(f"{product}: {count} purchases")
print("\n Inactive Customers:")
if inactive_customers:
print(", ".join(inactive_customers))
else:
print("No inactive customers found.")
print("\n AI Insights:")
for insight in ai_insights:
print(f"- {insight}")



Observation: 1.AI Assistance Used:

- **Helped design an intelligent logic flow using Python dictionaries and lists.**
- **Suggested AI-like reasoning to analyze data patterns, detect inactive users, and recommend marketing actions.**
- **Improved readability and maintainability with modular, structured code.**

2.Data Structure Logic:

- **Dictionary used to store both orders (order_data) and sales counts (product_sales).**
- **List comprehension used to find inactive customers efficiently.**
- **Sorting helps determine top-selling products.**

3.AI-Like Insights:

- **Promotes top-selling items.**
- **Suggests re-engagement offers to inactive users.**