***Project Report: Fractional Knapsack Application***

Student Name: **Piyush Jaloniya**  UID: **24MCI10042**

Student Name: **Khushi** UID: **24MCI10059**

Branch: MCA (AIML) Section/Group:24MAM 1-B

Semester: 1st Date: 06/11/2024

Subject Name: **Design and Analysis of Algorithms Labs** Subject Code: 24CAP-612

# Title

**Dynamic Inventory System with Knapsack-Based Profit Optimization**

# Aim

The aim of this project is to develop a dynamic inventory management system that optimizes profit by selectively choosing items for storage based on a fractional knapsack algorithm. This system is tailored to assist users in managing limited inventory space while maximizing potential profit from selected items.

# Objective/Problem Definition

In scenarios like e-commerce and retail, limited storage capacity and the need for profit maximization are critical challenges. This project addresses the problem by building an inventory management system that employs a fractional knapsack approach to prioritize high-profit items based on weight and value, thereby optimizing inventory utilization.

# Programming Languages Used

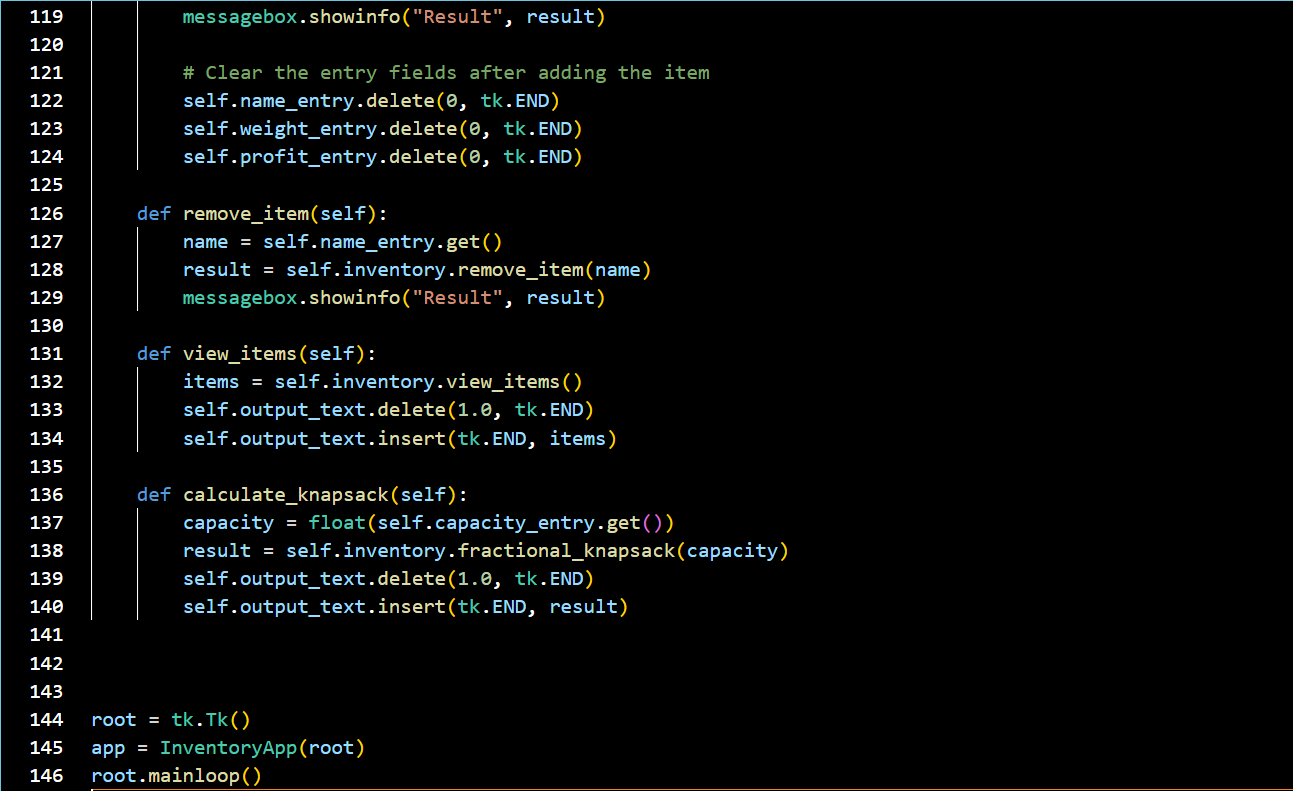
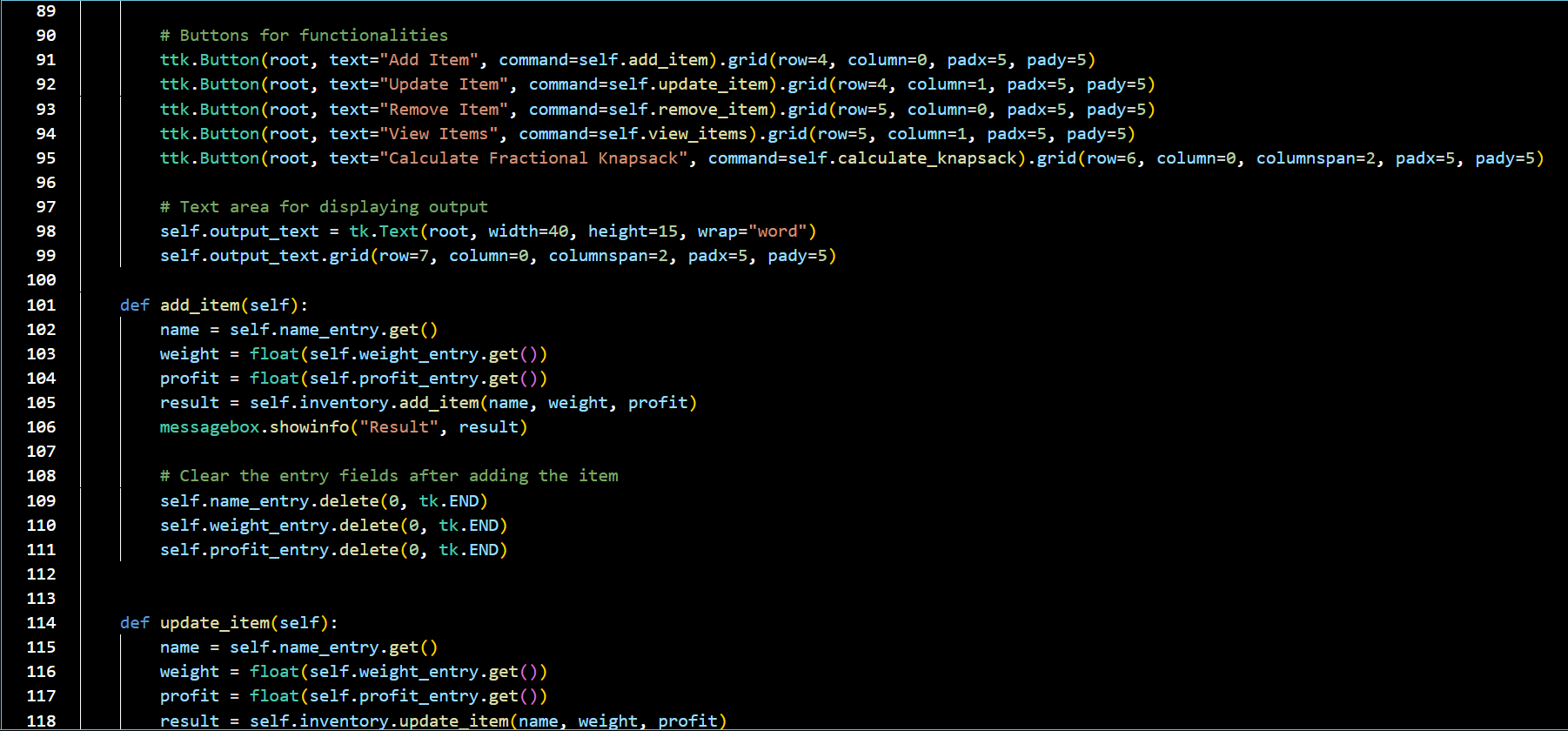
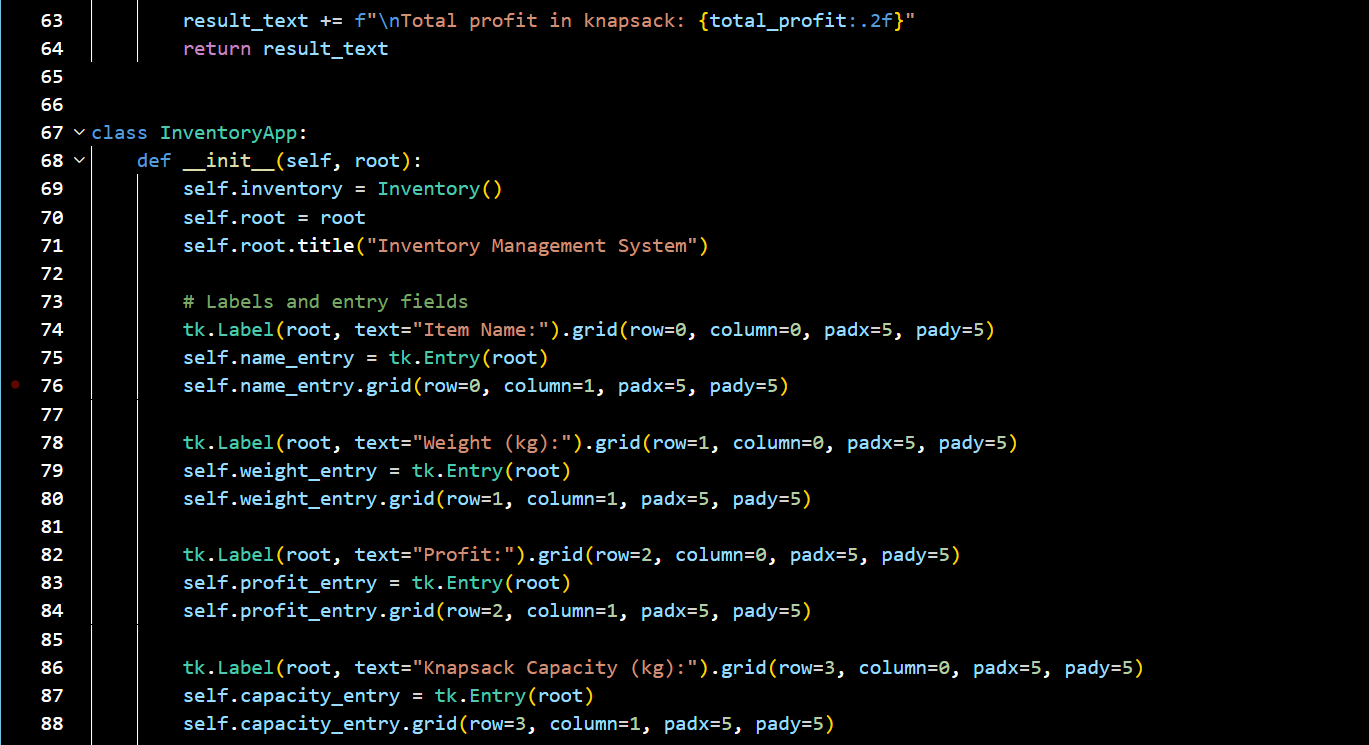
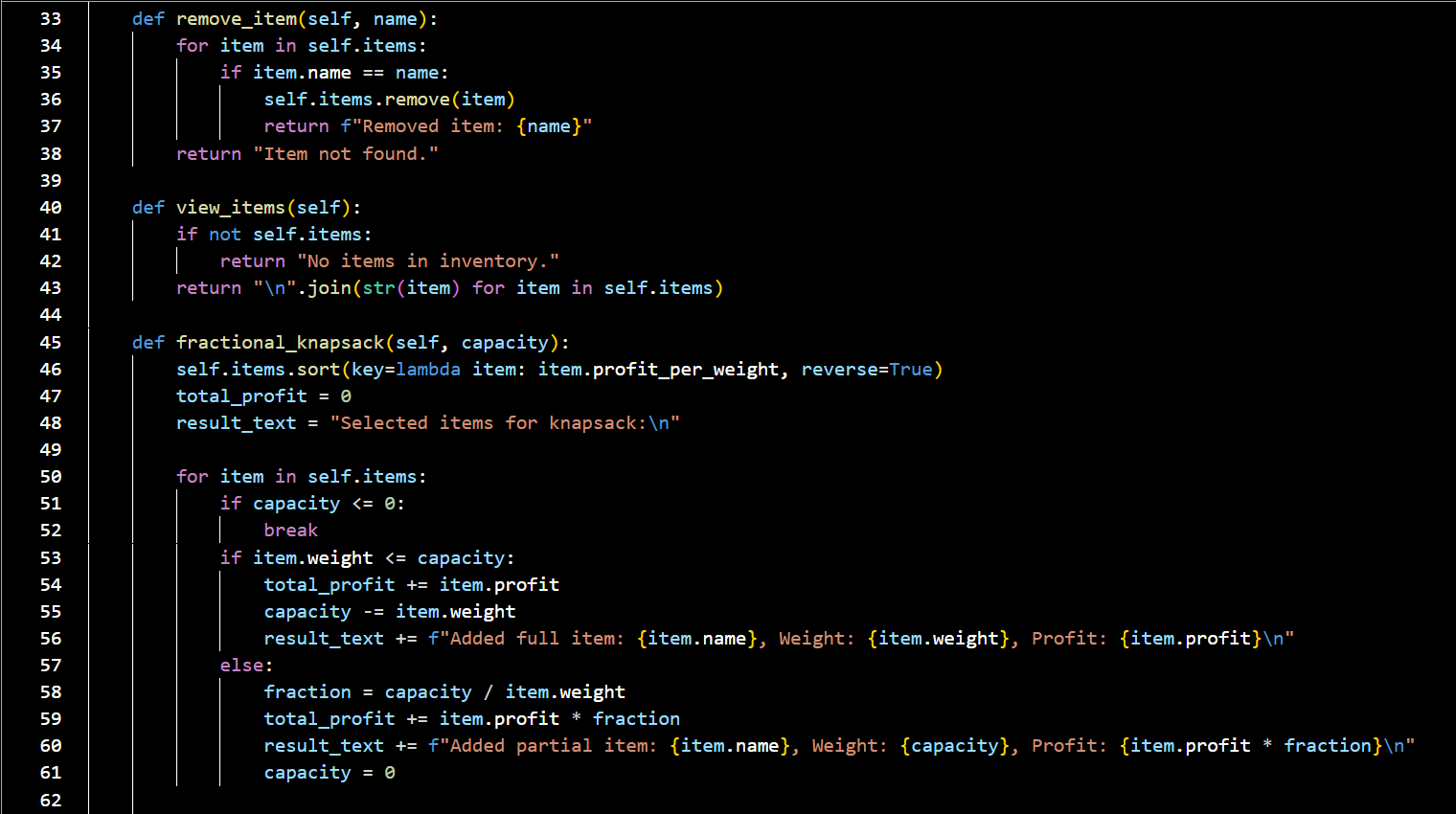
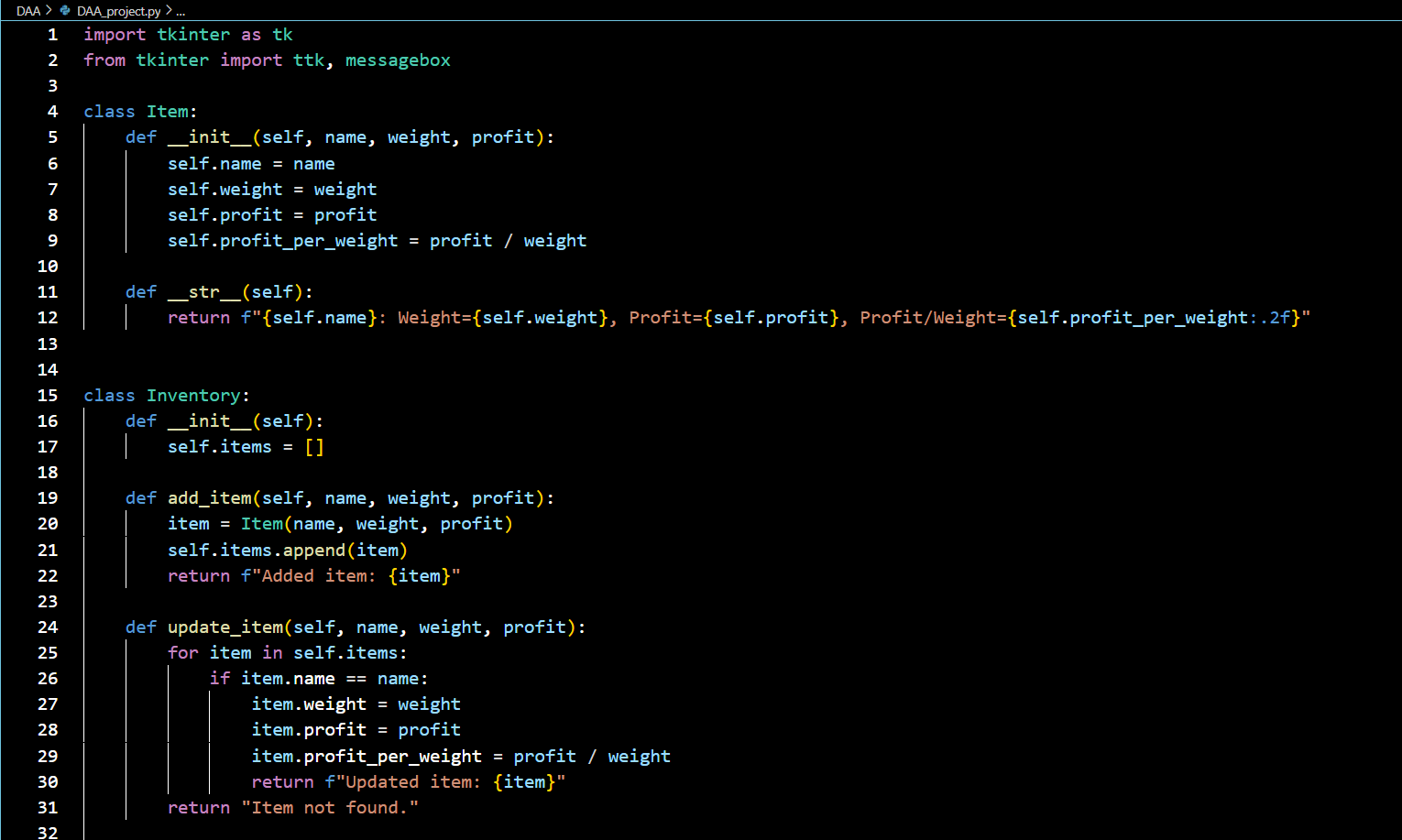
The project is implemented using Python, with the following libraries:  
- Tkinter: for creating a graphical user interface (GUI)  
- Standard Python libraries for data processing and handling the knapsack algorithm.

# Block Diagram/Design Flow/Flow Chart

The following flow chart illustrates the overall process of the Dynamic Inventory System:  
- Step 1: User inputs item details (name, weight, and profit) via the GUI.  
- Step 2: The item is added to the inventory, where it can be updated or removed.  
- Step 3: When required, the user enters a knapsack capacity.  
- Step 4: The knapsack algorithm calculates the maximum profit possible within the given capacity.

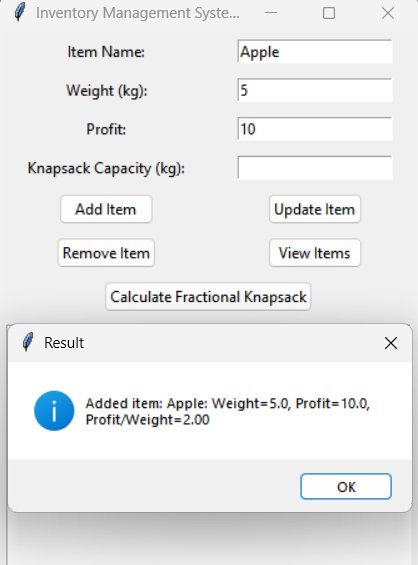
# Algorithm or Pseudo Code

Fractional Knapsack Algorithm:  
1. Sort items based on profit-to-weight ratio in descending order.  
2. For each item in the sorted list:  
 a. If the item's weight is less than or equal to the remaining capacity,  
 add its full profit to total profit and reduce capacity.  
 b. If not, add a fraction of its profit proportional to remaining capacity.  
3. Stop when no capacity remains.  
This approach maximizes profit within limited space by including high-value items.

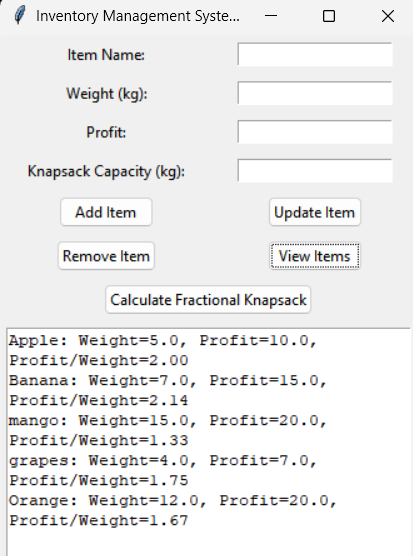
Implementation

# Output

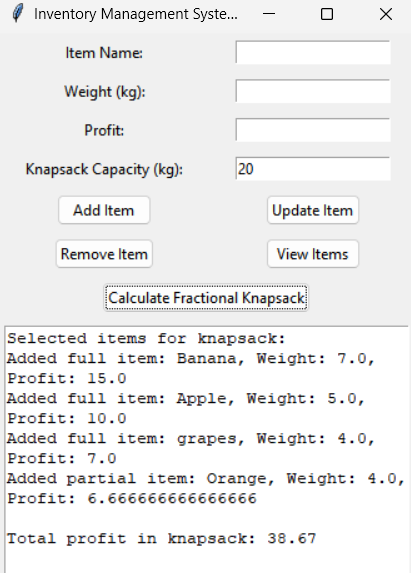
The application outputs include:  
**- Adding items: Displays a confirmation message.**



**- Viewing items: Lists all items in the inventory with details.**



**- Knapsack Calculation: Shows selected items for maximum profit and the total profit within capacity constraints.**



# Conclusion

The Dynamic Inventory System with Knapsack-Based Profit Optimization successfully demonstrates an efficient approach to inventory management. The system allows users to manage items, maximize profit, and make optimal use of limited storage capacity.

# Future Framework

Potential future improvements include:  
- Integrating a database for persistent storage  
- Enhancing the GUI for better usability  
- Adding advanced filtering and sorting options  
- Expanding the knapsack algorithm to handle complex constraints.

# Learning Outcomes

This project provided practical experience in:  
- Python programming and Tkinter for GUI applications  
- Applying the fractional knapsack algorithm for optimization  
- Understanding inventory management strategies in resource-limited settings