INVENTORY MANAGEMENT SYSTEM

N.S.Karthi

Aim of the Project

The aim of this project is to develop a simple yet effective inventory management system using Python dictionaries. This system will allow users to manage stock levels, add new items, and remove items as they are sold or used. The goal is to provide a foundational tool that can be expanded with more sophisticated features as needed.

Business Problem

Effective inventory management is crucial for businesses to avoid overstocking or understocking, which can lead to financial losses and inefficiencies. This project addresses the need for a straightforward, easy-to-use system to track and manage inventory in real-time, ensuring that businesses can maintain optimal stock levels and make informed decisions about purchasing and sales.

Project Description

This project implements an inventory management system using Python dictionaries. The system includes functionalities to add items to the inventory, update item quantities and prices, remove items, and display the current state of the inventory. The design is simple, focusing on core functionalities that are essential for managing inventory effectively.

Functionalities

1. Add Item: Allows users to add new items to the inventory or update existing items with new quantities and prices.

2. Remove Item: Enables users to remove a specified quantity of an item from the inventory. If the requested quantity exceeds the available quantity, all items are removed.

3. Get Inventory: Returns the current state of the inventory.

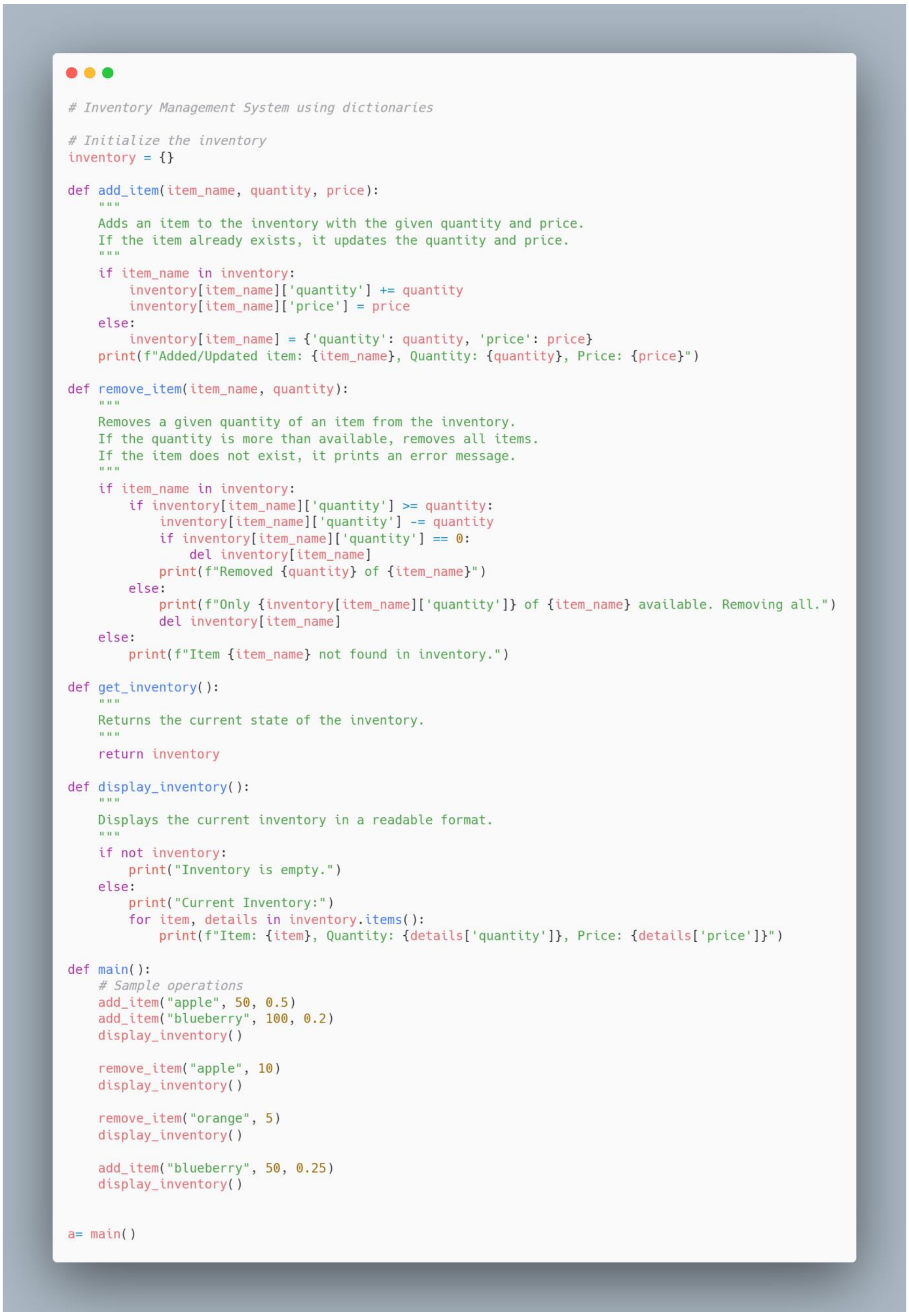
4. Display Inventory: Prints the current inventory in a readable format.

Input Versatility and Error Handling

Input Versatility: The system accepts string inputs for item names and ensures that quantities and prices are numeric. This ensures flexibility and accuracy in managing different types of inventory items.

Error Handling: The system includes basic error handling to manage scenarios such as attempting to remove more items than available or trying to remove items that do not exist in the inventory. Informative messages are printed to guide the user.

Code Implementation Description



```python
# Inventory Management System using dictionaries

# Initialize the inventory
inventory = {}

def add_item(item_name, quantity, price):
    """
    Adds an item to the inventory with the given quantity and price.
    If the item already exists, it updates the quantity and price.
    """
    if item_name in inventory:
        inventory[item_name]['quantity'] += quantity
        inventory[item_name]['price'] = price
    else:
        inventory[item_name] = {'quantity': quantity, 'price': price}
    print(f"Added/Updated item: {item_name}, Quantity: {quantity}, Price: {price}")

def remove_item(item_name, quantity):
    """
    Removes a given quantity of an item from the inventory.
    If the quantity is more than available, removes all items.
    If the item does not exist, it prints an error message.
    """
    if item_name in inventory:
        if inventory[item_name]['quantity'] >= quantity:
            inventory[item_name]['quantity'] -= quantity
            if inventory[item_name]['quantity'] == 0:
                del inventory[item_name]
            print(f"Removed {quantity} of {item_name}")
        else:
            print(f"Only {inventory[item_name]['quantity']} of {item_name} available. Removing all.")
            del inventory[item_name]
    else:
        print(f"Item {item_name} not found in inventory.")

def get_inventory():
    """
    Returns the current state of the inventory.
    """
    return inventory

def display_inventory():
    """
    Displays the current inventory in a readable format.
    """
    if not inventory:
        print("Inventory is empty.")
    else:
        print("Current Inventory:")
        for item, details in inventory.items():
            print(f"Item: {item}, Quantity: {details['quantity']}, Price: {details['price']}")

def main():
    # Sample operations
    add_item("apple", 50, 0.5)
    add_item("blueberry", 100, 0.2)
    display_inventory()

    remove_item("apple", 10)
    display_inventory()

    remove_item("orange", 5)
    display_inventory()

    add_item("blueberry", 50, 0.25)
    display_inventory()


a= main()
```

The inventory management system is implemented using Python dictionaries to manage and track items efficiently. The code is divided into several functions, each responsible for a specific aspect of inventory management.

1. Initialize the Inventory

python

```
inventory = {}
```

The inventory dictionary is initialized as an empty dictionary. This dictionary will store item names as keys and another dictionary as values. The inner dictionary contains quantity and price as keys.

2. Add Item to Inventory

python

```
def add_item(item_name, quantity, price):


  """

   Adds an item to the inventory with the given quantity and price.

   If the item already exists, it updates the quantity and price.

   """

  if item_name in inventory:
       inventory[item_name]['quantity'] += quantity
       inventory[item_name]['price'] = price
   else:
       inventory[item_name] = {'quantity': quantity, 'price': price}
   print(f"Added/Updated item: {item_name}, Quantity: {quantity}, Price: {price}")
```

This function adds a new item to the inventory or updates an existing item's quantity and price. It takes three parameters:

- item_name (str): The name of the item.

- quantity (int): The quantity of the item to add.

- price (float): The price of the item.

The function checks if the item already exists in the inventory:

- If it does, it increments the existing quantity and updates the price.

- If it doesn't, it creates a new entry in the inventory with the given quantity and price.

3. Remove Item from Inventory

python

```
def remove_item(item_name, quantity):
    """
    Removes a given quantity of an item from the inventory.
    If the quantity is more than available, removes all items.
    If the item does not exist, it prints an error message.
    """
    if item_name in inventory:
        if inventory[item_name]['quantity'] >= quantity:
            inventory[item_name]['quantity'] -= quantity
            if inventory[item_name]['quantity'] == 0:
                del inventory[item_name]
            print(f"Removed {quantity} of {item_name}")
        else:
            print(f"Only {inventory[item_name]['quantity']} of {item_name} available. Removing all.")
            del inventory[item_name]
    else:
        print(f"Item {item_name} not found in inventory.")
```

This function removes a specified quantity of an item from the inventory. It takes two parameters:

- item_name (str): The name of the item.

- quantity (int): The quantity of the item to remove.

The function checks if the item exists in the inventory:

- If it does, it checks if the requested quantity to remove is available.

- If the available quantity is sufficient, it subtracts the specified quantity

- If the available quantity is insufficient, it removes all quantities of the item.

- If the item does not exist, it prints an error message.

4. Get Inventory

```
python

def get_inventory():

    """

    Returns the current state of the inventory.

    """

    return inventory
```

This function returns the current state of the inventory dictionary. It doesn't take any parameters and simply returns the inventory dictionary.

5. Display Inventory

python

```
def display_inventory():
    """
    Displays the current inventory in a readable format.
    """
    if not inventory:
        print("Inventory is empty.")
    else:
        print("Current Inventory:")
        for item, details in inventory.items():
            print(f"Item: {item}, Quantity: {details['quantity']}, Price: {details['price']}")
```

This function prints the current state of the inventory in a readable format. It checks if the inventory is empty:

- If it is empty, it prints "Inventory is empty."

- If it is not empty, it prints each item with its quantity and price.

6. Main Function

python

```
def main():
    # Sample operations
    add_item("apple", 50, 0.5)
    add_item("banana", 100, 0.2)
    display_inventory()

    remove_item("apple", 10)
    display_inventory()

    remove_item("orange", 5)
    display_inventory()

    add_item("banana", 50, 0.25)
    display_inventory()
```

The main function demonstrates the functionality of the inventory management system with sample operations:

- Adding items to the inventory.

- Removing items from the inventory.

- Displaying the current inventory after each operation

7. Execution Entry Point

python

```
if __name__ == "__main__":
    main()
```

This statement ensures that the main function is called when the script is executed directly. It prevents the main function from running if the script is imported as a module in another script.

The provided code is a basic inventory management system using Python dictionaries. It allows users to add and remove items, update quantities and prices, and display the current inventory. The system includes basic error handling to manage common scenarios such as attempting to remove more items than available or trying to remove non-existent items. This implementation serves as a foundation for more advanced inventory management systems.

Results and Outcome

The implemented system successfully manages inventory by allowing users to add and remove items and view the current inventory status. The system is user-friendly, providing clear feedback and handling common errors gracefully. Sample operations demonstrate the functionality and effectiveness of the system.

```
Added/Updated item: apple, Quantity: 50, Price: 0.5
Added/Updated item: blueberry, Quantity: 100, Price: 0.2
Current Inventory:
Item: apple, Quantity: 50, Price: 0.5
Item: blueberry, Quantity: 100, Price: 0.2
Removed 10 of apple
Current Inventory:
Item: apple, Quantity: 40, Price: 0.5
Item: blueberry, Quantity: 100, Price: 0.2
Item orange not found in inventory.
Current Inventory:
Item: apple, Quantity: 40, Price: 0.5
Item: blueberry, Quantity: 100, Price: 0.2
Added/Updated item: blueberry, Quantity: 50, Price: 0.25
Current Inventory:
Item: apple, Quantity: 40, Price: 0.5
Item: blueberry, Quantity: 150, Price: 0.25
```

Conclusion

The inventory management system developed in this project serves as a fundamental tool for managing stock levels in a business. It provides essential functionalities needed to track and update inventory efficiently. The system can be further expanded with more advanced features such as automated restocking alerts, reporting capabilities, and integration with other business systems. This project highlights the potential of using Python dictionaries for simple and effective inventory management solutions.