



# K.RAMAKRISHNAN COLLEGE OF ENGINEERING(AUTONOMOUS), TRICHY

NAME	: P.DHARANI SHREE
YEAR	: 1 <sup>st</sup> YEAR “A” SECTION
REG NUMBER	: 8115U23EC018
SUBJECT	: PYTHON PROGRAMMING
ROLL NUMBER	: ECA2318
DEPARTMENT	: ELECTRONICS AND COMMUNICATION ENGINEERING
TOPIC	: INVENTORY MANAGEMENT SYSTEM FOR FRUIT AND GROCERY SHOP

# CONTENT:

- ❖ INTRODUCTION
- ❖ TECHNOLOGY[PYTHONLANGUAGE]
- ❖ WORKFLOW
- ❖ MODULECOMPLETION
- ❖ PROSANDCONS
- ❖ CONCLUSION

## INTRODUCTION:

In the dynamic environment of a bustling fruit and grocery shop, efficient inventory management stands as the backbone of seamless operations. Our inventory management system harnesses the power of Python to streamline the complexities of tracking, organizing, and replenishing a diverse range of products. From the vibrant array of fresh fruits to pantry staples, our system ensures real-time visibility into stock levels, enabling swift decision-making and optimized resource. By automating inventory tasks, such as product addition, depletion, and reordering, we empower shop owners to focus on delivering exceptional customer experiences while maintaining optimal stock levels to meet fluctuating demands.

# TECHNOLOGY USED:

1. **Point of Sale (POS) Systems:** POS systems are central to inventory management in retail environments. They handle transactions, track sales, and often integrate with inventory databases to automatically update stock levels in real-time.
2. **Barcode Scanners:** Barcode scanners are used to quickly and accurately scan product barcodes during checkout and inventory management processes. They help in identifying products, updating inventory levels, and reducing errors in data entry.
3. **Inventory Management Software:** Specialized inventory management software provides comprehensive tools for managing inventory, such as tracking stock levels, generating purchase orders, managing suppliers, and analyzing sales trends. These software solutions often offer features like batch tracking, expiry date tracking, and customizable reporting.
4. **Radio Frequency Identification (RFID):** RFID technology uses radio waves to identify and track items equipped with RFID tags. In grocery stores, RFID tags can be attached to products or pallets to enable automated tracking of inventory movement throughout the supply chain, from receiving to storage to checkout.

5. **Wireless Communication:** Wireless communication technologies like Wi-Fi and Bluetooth enable seamless data transfer between inventory management systems, POS terminals, handheld devices, and back-end databases. This facilitates real-time updates to inventory levels, pricing, and product information.

6. **Cloud Computing:** Cloud-based inventory management systems offer scalability, accessibility, and data redundancy. They allow multiple users to access inventory data from anywhere with an internet connection, enabling remote management, collaboration, and synchronization across multiple store locations.

7. **Data Analytics and Machine Learning:** Advanced analytics and machine learning techniques can be applied to inventory data to identify patterns, forecast demand, optimize stocking levels, and automate decision-making processes. These technologies help in reducing stockouts, minimizing overstocking, and improving inventory turnover ratios.

8. **Mobile Applications:** Mobile applications provide on-the-go access to inventory data, allowing store managers and staff to perform tasks such as stock counts, order processing, and price adjustments directly from smartphones or tablets.

# WORKFLOW:



## MODULE COMPLETION:

- ✓ AddItemstoInventory: Allow the user to add items to the inventory by providing the item name, quantity, and price.
- ✓ UpdateItemquantities: Allow the user to update the quantity of existing items in the inventory.
- ✓ RemoveItemsfromInventory: Provide functionality to remove items from the inventory based on the item name and quantity.
- ✓ DisplayInventory: Implement a function to display the current inventory with item names, quantities, and prices.
- ✓ Exit : **Exit from the system of inventory management.**

# PROS:

- **Simplicity and Readability:** Python's clean and readable syntax makes it easy to develop and maintain code, reducing development time and effort.
- **Vast Ecosystem of Libraries:** Python offers a rich selection of libraries for tasks such as data manipulation, web development, and database integration, providing ready-made solutions for common inventory management challenges.
- **Cross-platform Compatibility:** Python is platform-independent, allowing the inventory management system to run on various operating systems without modification.
- **Community Support:** Python has a large and active community of developers who contribute to forums, tutorials, and open-source projects, providing valuable resources and assistance for troubleshooting.



## CONS:

- **Complexity:** Depending on the scale and requirements of the shop, implementing a comprehensive inventory management system can be quite complex.
- **Data Accuracy:** Ensuring the accuracy of inventory data can be challenging, especially in a fast-paced environment where items are bought, sold, and restocked frequently.
- **Concurrency:** In a multi-user environment, concurrent access to the inventory system can lead to issues such as race conditions, where multiple users or processes attempt to modify the same inventory data simultaneously.
- **Performance:** As the size of the inventory grows, the performance of the system may degrade, particularly in operations such as searching, updating, and generating reports.

## CONCLUSION:

In conclusion, developing an inventory management system for fruit and grocery stores in Python presents both opportunities and challenges. By leveraging Python's flexibility and extensive libraries, you can create a robust system tailored to the specific needs of the business. However, it's essential to address various considerations to ensure the system's effectiveness and reliability.

## PROGRAM:

```
class InventoryItem:
```

```
    def __init__(self, name, quantity, price):
```

```
        self.name = name
```

```
        self.quantity = quantity
```

```
        self.price = price
```

```
    def display_info(self):
```

```
        print(f"Name: {self.name}, Quantity: {self.quantity}, Price:  
${self.price:.2f}")
```

```
class InventoryManagementSystem:
```

```
    def __init__(self):
```

```
        self.inventory = {}
```

```
def add_item(self, item):  
    self.inventory[item.name] = item  
  
def remove_item(self, name):  
    if name in self.inventory:  
        del self.inventory[name]  
  
def update_item_quantity(self, name, quantity):  
    if name in self.inventory:  
        self.inventory[name].quantity += quantity  
  
def update_item_price(self, name, price):  
    if name in self.inventory:  
        self.inventory[name].price = price  
  
def display_inventory(self):
```

```
print("Inventory:")  
for item in self.inventory.values():  
    item.display_info()
```

```
def main():  
    inventory_system = InventoryManagementSystem()
```

```
    while True:  
        print("\n1. Add Item")  
        print("2. Remove Item")  
        print("3. Update Quantity")  
        print("4. Update Price")  
        print("5. Display Inventory")  
        print("6. Exit")  
        choice = input("Enter your choice: ")
```

```
if choice == "1":
    name = input("Enter the name of the item: ")
    quantity = int(input("Enter the quantity: "))
    price = float(input("Enter the price per unit: "))
    item = InventoryItem(name, quantity, price)
    inventory_system.add_item(item)
    print(f"{name} added to the inventory.")
elif choice == "2":
    name = input("Enter the name of the item to remove: ")
    inventory_system.remove_item(name)
    print(f"{name} removed from the inventory.")
elif choice == "3":
    name = input("Enter the name of the item to update quantity: ")
    quantity = int(input("Enter the quantity to add/remove: "))
    inventory_system.update_item_quantity(name, quantity)
elif choice == "4":
    name = input("Enter the name of the item to update price: ")
```

```
    price = float(input("Enter the new price: "))
    inventory_system.update_item_price(name, price)
elif choice == "5":
    inventory_system.display_inventory()
elif choice == "6":
    print("Exiting Inventory Management System.")
    break
else:
    print("Invalid choice. Please enter a valid option.")
```

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

## OUTPUT:

```
$ python CTP28132.py puthonpython ython k

1. Add Item
2. Remove Item
3. Update Quantity
4. Update Price
5. Display Inventory
6. Exit
Enter your choice: y
Invalid choice. Please enter a valid option.

1. Add Item
2. Remove Item
3. Update Quantity
4. Update Price
5. Display Inventory
6. Exit
```



THANKYOU!!