MEDICINE TRACKING SYSTEM

PRESENTED BY

Name: Neha Spriha Baruah

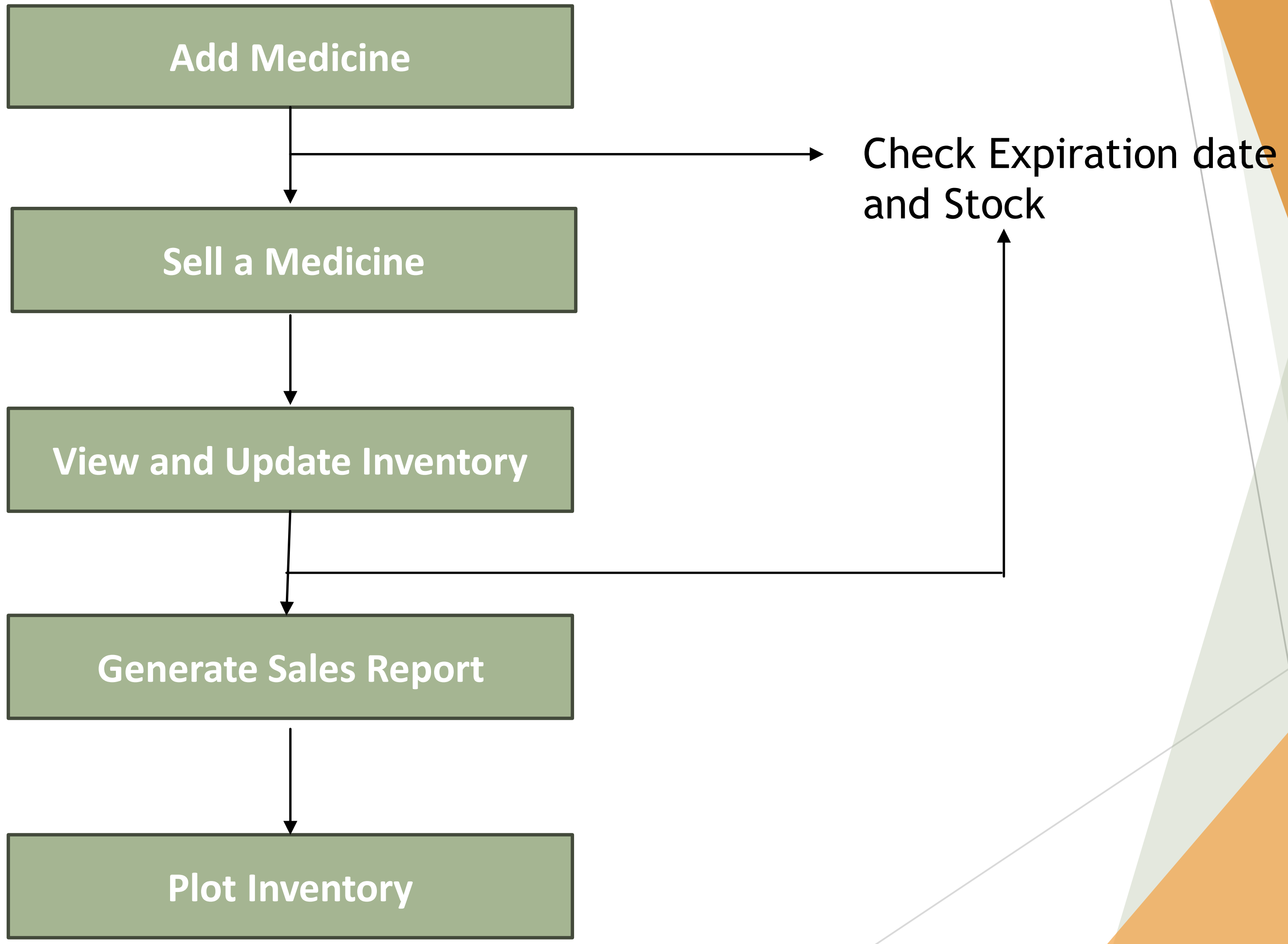
Roll no: 246102012

Subject Code: DA514

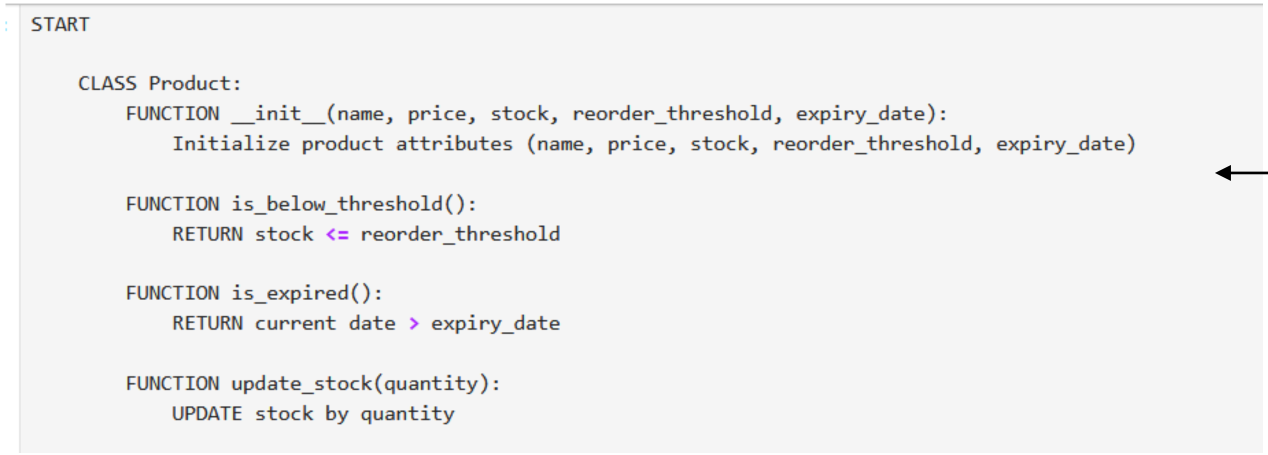
MOTIVATION

- **Inventory Management and Stock Optimization**
  - ➢ Prevent Stockouts and Overstocks
  - ➢ Expiration Tracking
  - ➢ Automatic Reordering
- **Improved Accuracy in Sales and Billing**
  - ➢ Efficient Sales Tracking
- **Enhanced Customer Service**
  - ➢ Quick Access to Information
- **Compliance and Regulatory Adherence**
  - ➢ Record-Keeping for Audits
- **Cost and Time Efficiency**
  - ➢ Reduced Operational Costs and time savings

BLOCK DIAGRAM

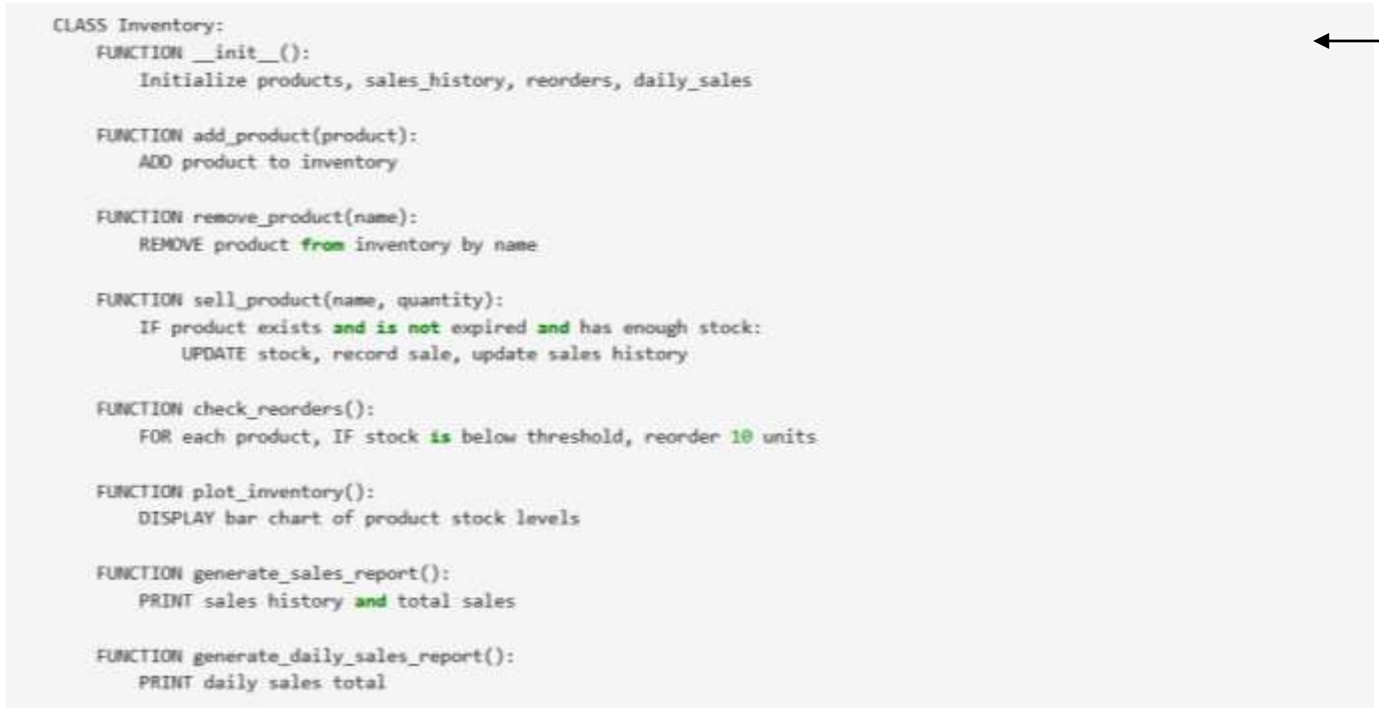


Pseudo Code



```
START

    CLASS Product:
        FUNCTION __init__(name, price, stock, reorder_threshold, expiry_date):
            Initialize product attributes (name, price, stock, reorder_threshold, expiry_date)

        FUNCTION is_below_threshold():
            RETURN stock <= reorder_threshold

        FUNCTION is_expired():
            RETURN current date > expiry_date

        FUNCTION update_stock(quantity):
            UPDATE stock by quantity
```

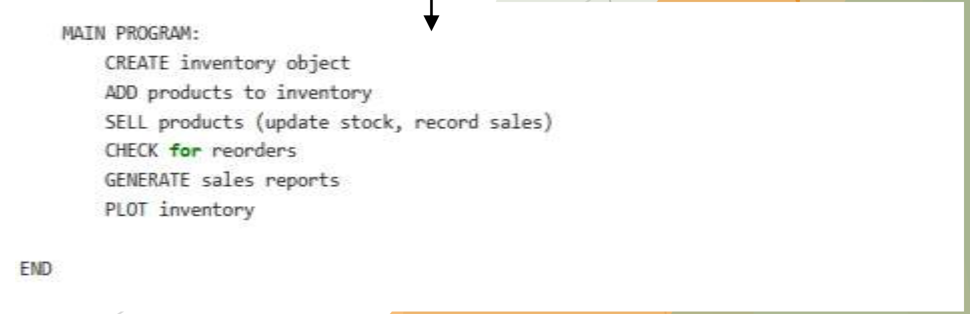
1. CLASS Product created



```
CLASS Inventory:
    FUNCTION __init__():
        Initialize products, sales_history, reorders, daily_sales

    FUNCTION add_product(product):
        ADD product to inventory

    FUNCTION remove_product(name):
        REMOVE product from inventory by name

    FUNCTION sell_product(name, quantity):
        IF product exists and is not expired and has enough stock:
            UPDATE stock, record sale, update sales history

    FUNCTION check_reorders():
        FOR each product, IF stock is below threshold, reorder 10 units

    FUNCTION plot_inventory():
        DISPLAY bar chart of product stock levels

    FUNCTION generate_sales_report():
        PRINT sales history and total sales

    FUNCTION generate_daily_sales_report():
        PRINT daily sales total
```

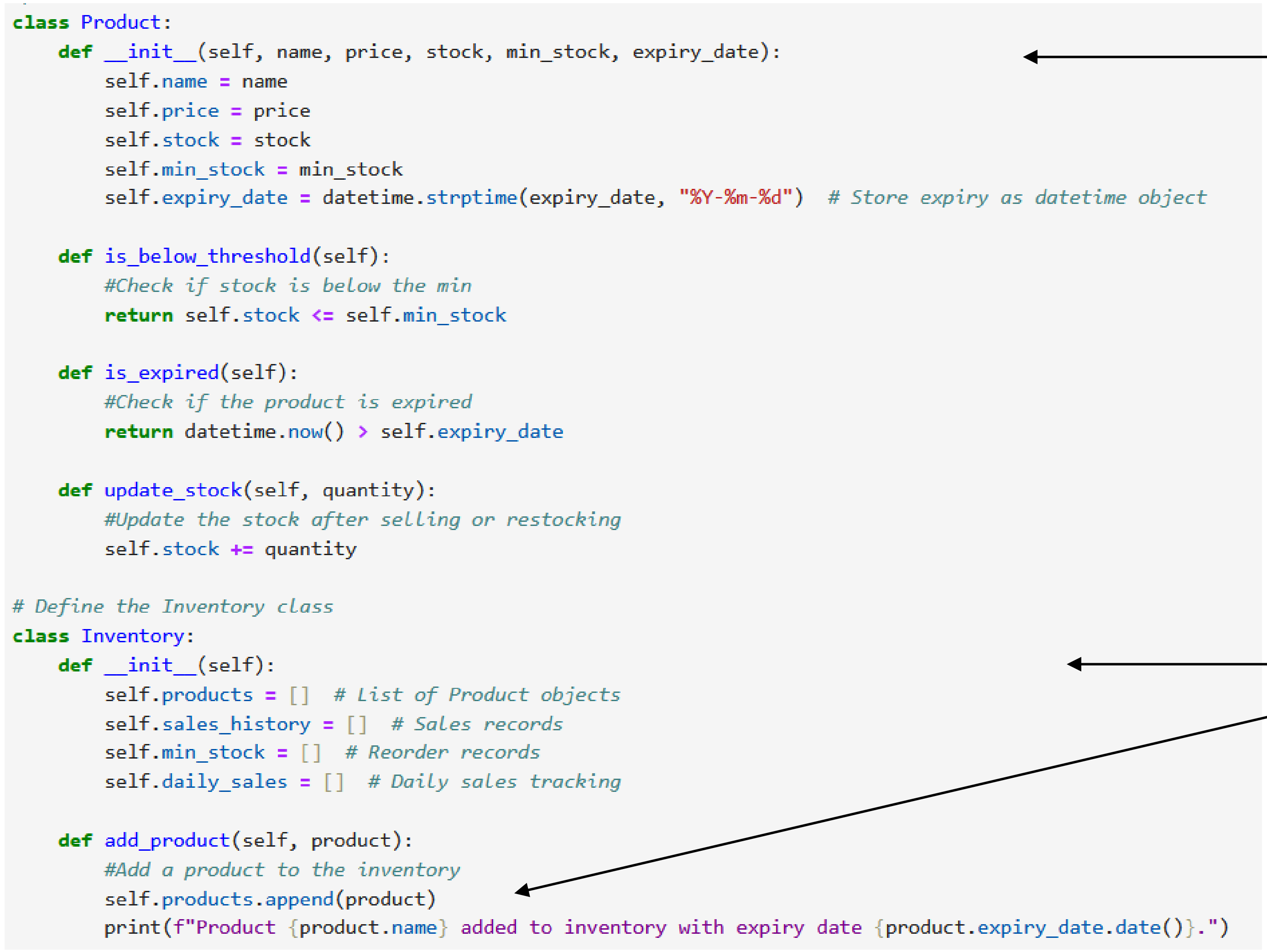
2. CLASS Inventory created

3. Main Function



```
    MAIN PROGRAM:
        CREATE inventory object
        ADD products to inventory
        SELL products (update stock, record sales)
        CHECK for reorders
        GENERATE sales reports
        PLOT inventory

END
```

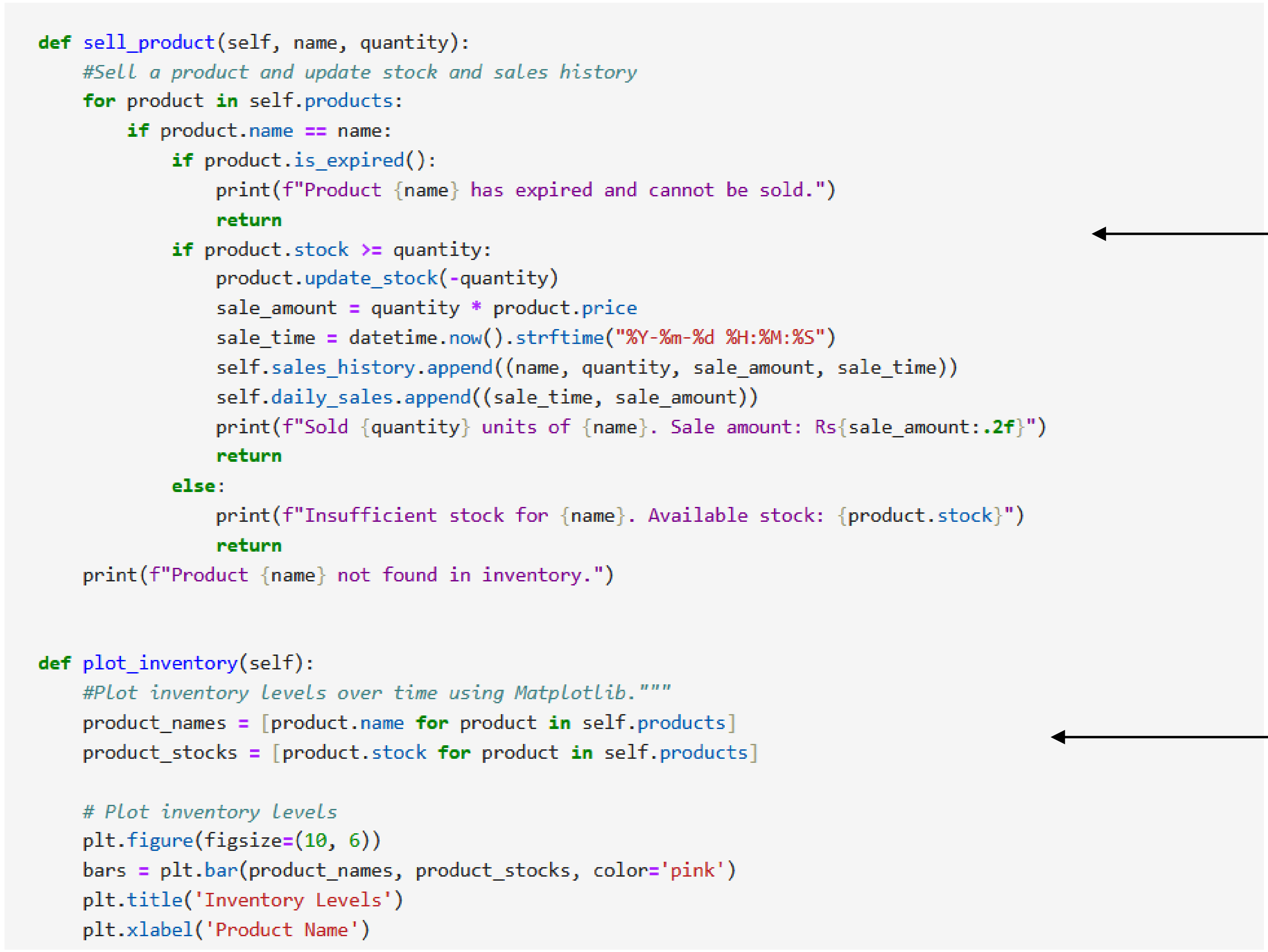
Python Code Snippets



```python
class Product:
    def __init__(self, name, price, stock, min_stock, expiry_date):
        self.name = name
        self.price = price
        self.stock = stock
        self.min_stock = min_stock
        self.expiry_date = datetime.strptime(expiry_date, "%Y-%m-%d")  # Store expiry as datetime object

    def is_below_threshold(self):
        #Check if stock is below the min
        return self.stock <= self.min_stock

    def is_expired(self):
        #Check if the product is expired
        return datetime.now() > self.expiry_date

    def update_stock(self, quantity):
        #Update the stock after selling or restocking
        self.stock += quantity

# Define the Inventory class
class Inventory:
    def __init__(self):
        self.products = []  # List of Product objects
        self.sales_history = []  # Sales records
        self.min_stock = []  # Reorder records
        self.daily_sales = []  # Daily sales tracking

    def add_product(self, product):
        #Add a product to the inventory
        self.products.append(product)
        print(f"Product {product.name} added to inventory with expiry date {product.expiry_date.date()}.")
```

Creating Classes and Defining Functions

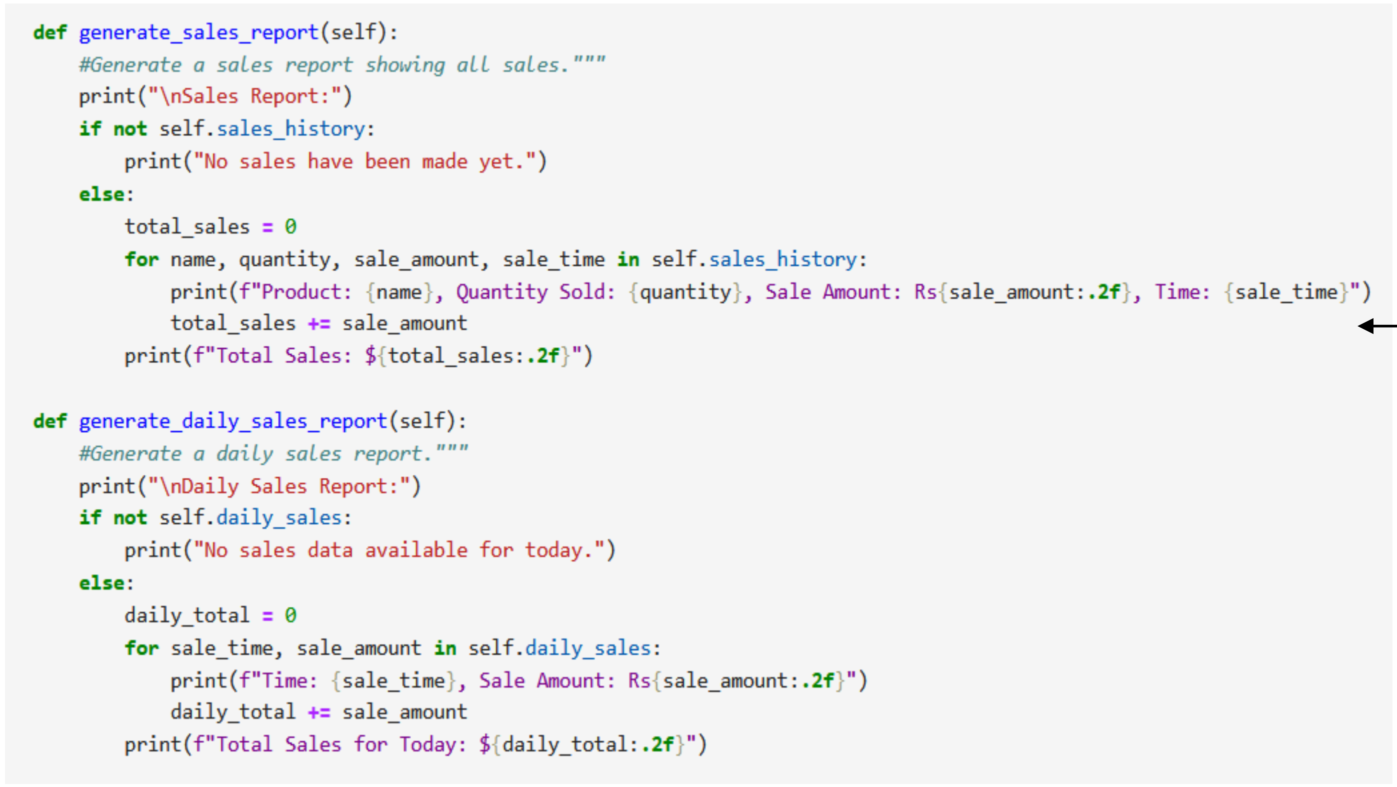
Creating and Accessing Lists



```python
def sell_product(self, name, quantity):
    #Sell a product and update stock and sales history
    for product in self.products:
        if product.name == name:
            if product.is_expired():
                print(f"Product {name} has expired and cannot be sold.")
                return
            if product.stock >= quantity:
                product.update_stock(-quantity)
                sale_amount = quantity * product.price
                sale_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                self.sales_history.append((name, quantity, sale_amount, sale_time))
                self.daily_sales.append((sale_time, sale_amount))
                print(f"Sold {quantity} units of {name}. Sale amount: Rs{sale_amount:.2f}")
                return
            else:
                print(f"Insufficient stock for {name}. Available stock: {product.stock}")
                return
    print(f"Product {name} not found in inventory.")


def plot_inventory(self):
    #Plot inventory levels over time using Matplotlib."""
    product_names = [product.name for product in self.products]
    product_stocks = [product.stock for product in self.products]

    # Plot inventory levels
    plt.figure(figsize=(10, 6))
    bars = plt.bar(product_names, product_stocks, color='pink')
    plt.title('Inventory Levels')
    plt.xlabel('Product Name')
```

Creating Loops and using of if-else statements

Plotting of Graphs using Matlabplot library



```python
def generate_sales_report(self):
    #Generate a sales report showing all sales."""
    print("\nSales Report:")
    if not self.sales_history:
        print("No sales have been made yet.")
    else:
        total_sales = 0
        for name, quantity, sale_amount, sale_time in self.sales_history:
            print(f"Product: {name}, Quantity Sold: {quantity}, Sale Amount: Rs{sale_amount:.2f}, Time: {sale_time}")
            total_sales += sale_amount
        print(f"Total Sales: ${total_sales:.2f}")

def generate_daily_sales_report(self):
    #Generate a daily sales report."""
    print("\nDaily Sales Report:")
    if not self.daily_sales:
        print("No sales data available for today.")
    else:
        daily_total = 0
        for sale_time, sale_amount in self.daily_sales:
            print(f"Time: {sale_time}, Sale Amount: Rs{sale_amount:.2f}")
            daily_total += sale_amount
        print(f"Total Sales for Today: ${daily_total:.2f}")
```

Using of loops and If not statements

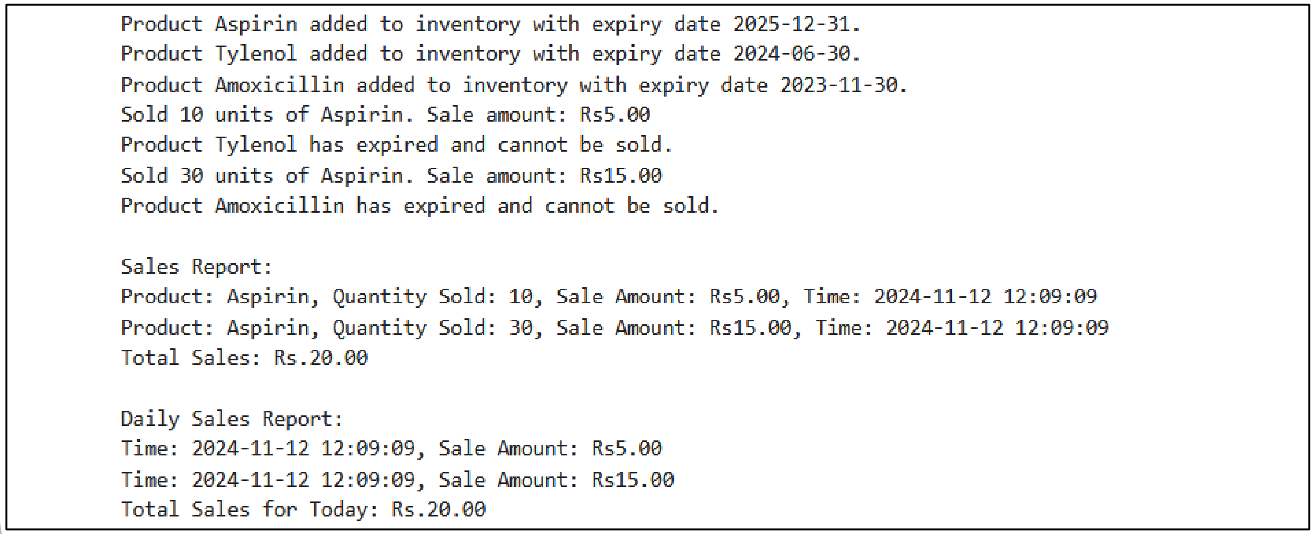
Results

INPUT: (ADD PRODUCTS)

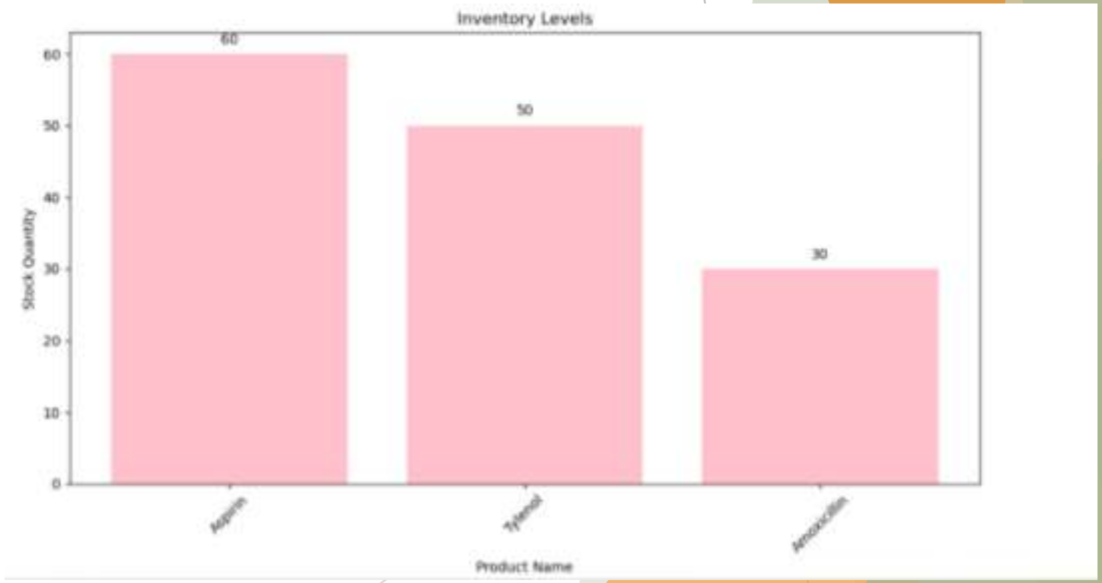
Drug: Aspirin , Tylenol, Amoxicillin
Price: Rs.0.50, Rs. 1.00, Rs. 2.00
Stock:100 , 50, 30
Min.Stock: 20, 10,10
Expiration date: 2025-12-31, 2024-06-30, 2023-11-30

INPUT: (SELL PRODUCTS)

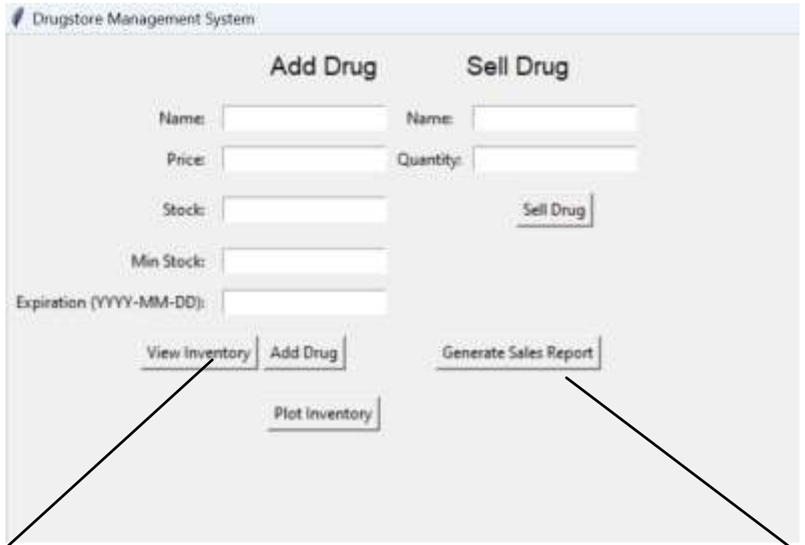
Drug: Aspirin , Tylenol, Amoxicillin
Sold: 10+30, 5, 2

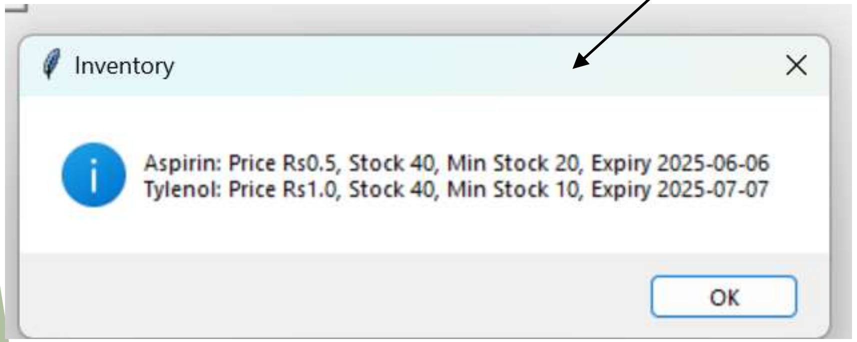


```
Product Aspirin added to inventory with expiry date 2025-12-31.
Product Tylenol added to inventory with expiry date 2024-06-30.
Product Amoxicillin added to inventory with expiry date 2023-11-30.
Sold 10 units of Aspirin. Sale amount: Rs5.00
Product Tylenol has expired and cannot be sold.
Sold 30 units of Aspirin. Sale amount: Rs15.00
Product Amoxicillin has expired and cannot be sold.

Sales Report:
Product: Aspirin, Quantity Sold: 10, Sale Amount: Rs5.00, Time: 2024-11-12 12:09:09
Product: Aspirin, Quantity Sold: 30, Sale Amount: Rs15.00, Time: 2024-11-12 12:09:09
Total Sales: Rs.20.00

Daily Sales Report:
Time: 2024-11-12 12:09:09, Sale Amount: Rs5.00
Time: 2024-11-12 12:09:09, Sale Amount: Rs15.00
Total Sales for Today: Rs.20.00
```

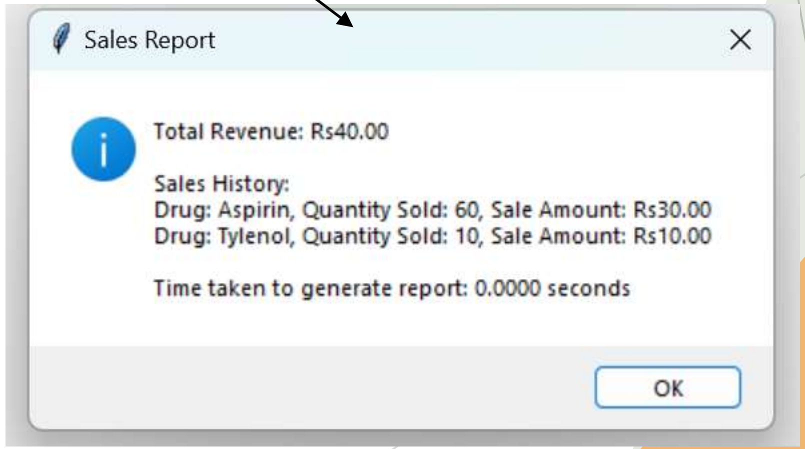


Graphical User Interface





Aspirin: Price Rs0.5, Stock 40, Min Stock 20, Expiry 2025-06-06
Tylenol: Price Rs1.0, Stock 40, Min Stock 10, Expiry 2025-07-07



```
Total Revenue: Rs40.00

Sales History:
Drug: Aspirin, Quantity Sold: 60, Sale Amount: Rs30.00
Drug: Tylenol, Quantity Sold: 10, Sale Amount: Rs10.00

Time taken to generate report: 0.0000 seconds
```

Conclusion

- Effective use of Basic concepts of python
- Simplifying codes using OOP and functions
- Accessing Lists and Loops in a code
- Plotting of Graph from data
- Graphical User Interface