Coding for a small retail store's inventory management system

by Basal, Cruz, and Flores

Exercise 5: Inventory Management System

Scenario: You are tasked with developing an inventory management system for a small retail store. The store needs to keep track of products, their quantities, and pricing.

Task 1: Write a function find_max_price(products) that finds the most expensive product in the inventory.

Task 2: Write a function find_min_stock(products) that identifies the product with the lowest stock, so the store can reorder it.

Task 3: Write a function average_price(products) that calculates the average price of all products to help with pricing strategies.

Task 4: Write a function remove_out_of_stock(products) that removes products with zero stock from the inventory list.

Task 5: Create a main program that simulates the store's inventory management process. Allow the user (store manager) to input product details, and then use the functions to analyze and manage the inventory, providing insights such as the most expensive item, average pricing, and stock status.

In [6]:
```
#Defining the functions

# Function to find the most expensive product
def find_max_price(products):

    # Check if the products list is empty
    if not products:
        return None

    # Find the product with the maximum price
    max_product = max(products, key=lambda x: x['price'])
    return max_product


# Function to find the product with the lowest stock
def find_min_stock(products):

    # Check if the products list is empty
    if not products:
        return None

    # Find the product with the minimum quantity
    min_stock_product = min(products, key=lambda x: x['quantity'])
    return min_stock_product


# Function to calculate the average price of products
def average_price(products):

    # Check if the products list is empty
    if not products:
        return 0

    # Sum up the prices of all products
    total_price = sum(product['price'] for product in products)

    # Calculate the average price
    avg_price = total_price / len(products)
    return avg_price


# Function to remove products with zero stock
def remove_out_of_stock(products):

    # Create a new list excluding products with zero quantity
    return [product for product in products if product['quantity'] > 0]



# Main program to manage the inventory
def main():

    # Initialize an empty list to store product information
    products = []

    while True:
        # Display the menu options
        print("\n--- Inventory Management System ---")
        print("1. Add a new product")
        print("2. Find the most expensive product")
        print("3. Find the product with the lowest stock")
        print("4. Calculate the average price of products")
        print("5. Remove out-of-stock products")
        print("6. Display current inventory")
        print("7. Exit")

        # Get user's choice from the menu
        choice = input("Enter your choice (1-7): ")

        # Option 1 to add a new product to the inventory
        if choice == '1':

            name = input("Enter product name: ")
            price = float(input("Enter product price: "))
            quantity = int(input("Enter product quantity: "))

            # Append the new product to the products list
            products.append({'name': name, 'price': price, 'quantity': quantity})
            print(f"Product {name} added to inventory.")

            # Display current inventory after adding the new product
            print("Current Inventory:")
            for product in products:
                print(f"{product['name']} - Price: ₱{product['price']:.2f}, Quantity: {product['quantity']} units")
                # Note that product price returns in ₱ currency and in 2 decimal places, and quantity in number of units

        # Option 2 to find and display the most expensive product
        elif choice == '2':

            max_product = find_max_price(products)

            if max_product:
                print(f"The most expensive product is {max_product['name']} with a price of ₱{max_product['price']:.2f}.")
                # Note that product price returns in ₱ currency and in 2 decimal places

            else:
                print("Inventory is empty.")

        # Option to find and display the product with the lowest stock
        elif choice == '3':

            min_stock_product = find_min_stock(products)

            if min_stock_product:
                print(f"The product with the lowest stock is {min_stock_product['name']} with {min_stock_product['quantity']} units left.")
                # Note that product quantity returns in number of units

            else:
                print("Inventory is empty.")

        # Option to calculate and display the average price of products
        elif choice == '4':

            avg_price = average_price(products)
            print(f"The average price of products is ₱{avg_price:.2f}.")
            # Note that product price returns in ₱ currency and in 2 decimal places

        # Option to remove products with zero stock from the inventory
        elif choice == '5':
            products = remove_out_of_stock(products)
            print("Out-of-stock products have been removed.")

        # Option to display the current inventory
        elif choice == '6':

            if products:
                print("Current Inventory:")
                for product in products:
                    print(f"{product['name']} - Price: ₱{product['price']:.2f}, Quantity: {product['quantity']} units")
                    # Note that product price returns in ₱ currency and in 2 decimal places, and quantity in number of units

            else:
                print("Inventory is empty.")

        # Option to exit the program
        elif choice == '7':

            print("Exiting Inventory Management System.")
            break

        # Handle invalid menu choices
        else:

            print("Invalid choice. Please select a valid option.")

# Entry point of the program
if __name__ == "__main__":
    main()
```

```
--- Inventory Management System ---
1. Add a new product
2. Find the most expensive product
3. Find the product with the lowest stock
4. Calculate the average price of products
5. Remove out-of-stock products
```

```
6. Display current inventory
7. Exit
Exiting Inventory Management System.
```

In [ ]: