LU03.A05 - Verwaltung von Lagerbeständen

Erstellen Sie ein Lagerverwaltungssystem, das verschiedene Berechnungen auf Lagerbeständen durchführt. Verwenden Sie Higher-Order Functions, um die Berechnungen modular und anpassbar zu gestalten.

Detaillierte Aufgabenstellung

Sie sind für die Verwaltung der Lagerbestände eines kleinen Unternehmens verantwortlich. Sie müssen ein System entwickeln, das verschiedene Berechnungen auf den Lagerbeständen durchführt. Halten Sie sich an die Kommentare in der Vorlage um die Funktionen zu erstellen.

Vorlage

```
def manage_inventory(products, calculation_function, *args):
    """
    Diese Funktion ist eine Higher-Order Funktion, die eine Liste von Produkten und eine Berechnungsfunktion akzeptiert.
    Die Berechnungsfunktion wird auf die Liste der Produkte angewendet, um eine spezifische Berechnung durchzuführen,
    z.B. den Gesamtwert des Lagerbestands berechnen oder Produkte unter einem bestimmten Bestand filtern.

    :param products: Liste von Produkten, jedes Produkt ist ein Dictionary mit 'name', 'price' und 'stock'
    :param calculation_function: Die Funktion, die auf die Produktliste angewendet wird
    :param *args: Weitere Argumente, die von der Berechnungsfunktion benötigt werden könnten
    :return: Die Berechnungsfunktion mit den Produkten und allenfalls zusätzlichen Argumente
    """


def calculate_average_price(products):
    """
    Diese Funktion berechnet den durchschnittlichen Preis der Produkte im Lagerbestand.

    :param products: Liste von Produkten
    :return: Durchschnittspreis der Produkte
    """
```

```
def calculate_total_value(products):
    """
    Diese Funktion berechnet den Gesamtwert der Produkte im Lagerbestand.
    Der Gesamtwert wird berechnet, indem der Preis jedes Produkts mit seinem
Bestand multipliziert
    und dann die Werte aller Produkte summiert werden.

    :param products: Liste von Produkten, jedes Produkt ist ein Dictionary
mit 'name', 'price' und 'stock'
    :return: Gesamtwert der Produkte im Lagerbestand
    """



def filter_products_by_stock(products, min_stock):
    """
    Diese Funktion filtert Produkte aus der Liste, basierend auf einem
minimalen Bestandskriterium.
    Nur Produkte mit einem Bestand, der gleich oder größer als der
angegebene min_stock ist,
    werden in die Ergebnisliste aufgenommen.

    :param products: Liste von Produkten, jedes Produkt ist ein Dictionary
mit 'name', 'price' und 'stock'
    :param min_stock: Die minimale Bestandsmenge, die ein Produkt haben
muss, um in die Ergebnisliste aufgenommen zu werden
    :return: Liste der gefilterten Produkte
    """



if __name__ == '__main__':
    # Produktliste: Eine Liste von Produkten, wobei jedes Produkt ein
Dictionary mit den Eigenschaften 'name', 'price' und 'stock' ist.
    products = [
        {'name': 'Laptop', 'price': 800, 'stock': 5},
        {'name': 'Smartphone', 'price': 500, 'stock': 10},
        {'name': 'Headphones', 'price': 50, 'stock': 20},
        {'name': 'Keyboard', 'price': 20, 'stock': 30},
        {'name': 'Monitor', 'price': 150, 'stock': 15},
        {'name': 'Mouse', 'price': 10, 'stock': 40},
        {'name': 'Printer', 'price': 200, 'stock': 10},
        {'name': 'Tablet', 'price': 300, 'stock': 12},
        {'name': 'Desk Chair', 'price': 100, 'stock': 7},
        {'name': 'USB Drive', 'price': 5, 'stock': 100},
        {'name': 'External Hard Drive', 'price': 80, 'stock': 20},
        {'name': 'Microphone', 'price': 40, 'stock': 25},
        {'name': 'Webcam', 'price': 30, 'stock': 15},
        {'name': 'Projector', 'price': 400, 'stock': 8},
        {'name': 'Speaker', 'price': 35, 'stock': 30},
```

```
        {'name': 'Smartwatch', 'price': 150, 'stock': 14},
        {'name': 'Phone Charger', 'price': 10, 'stock': 50},
        {'name': 'Laptop Bag', 'price': 25, 'stock': 30},
        {'name': 'HDMI Cable', 'price': 10, 'stock': 40},
        {'name': 'WiFi Router', 'price': 60, 'stock': 12}
    ]

    # Test: Berechnung des durchschnittlichen Preises
    average_price = manage_inventory(products, calculate_average_price)
    print(f'Durchschnittlicher Preis der Produkte: {average_price:.2f}€')

    # Test: Berechnung des Gesamtwerts
    total_value = manage_inventory(products, calculate_total_value)
    print(f'Gesamtwert der Produkte: {total_value}€')

    # Test: Filtern von Produkten mit einem minimalen Bestand von 20
    min_stock = 20
    filtered_products = manage_inventory(products, filter_products_by_stock,
min_stock)
    print(f'Produkte mit einem Bestand von mindestens {min_stock}:')
    for product in filtered_products:
        print(product)

    average_price_min_stock = manage_inventory(filtered_products,
calculate_average_price)
    print(f'Durchschnittlicher Preis der Produkte wo min_stock=20:
{average_price_min_stock:.2f}€')
```
