LU02.L06 - Einkaufsliste

Funktion zur Preisanpassung:

```
def update_prices(products, new_prices):
    updated_products = []
    for product in products:
        updated_product = product.copy()
        product_name = updated_product['product']
        if product_name in new_prices:
            updated_product['price'] = new_prices[product_name]
        updated_products.append(updated_product)
    return updated_products
```

Vorgehen

In der Funktion zur Preisanpassung besteht die Hauptaufgabe darin, die Preise in der Einkaufsliste basierend auf den neuen Preisen zu aktualisieren, ohne die ursprüngliche Liste zu verändern.

1. Erstellen einer neuen leeren Liste: Diese Liste wird verwendet, um die aktualisierten Produkte zu speichern, ohne die ursprüngliche Liste zu ändern.
2. Durchlaufen der Produkte in der ursprünglichen Liste: Für jedes Produkt im ursprünglichen Produkt wird eine Kopie erstellt, um sicherzustellen, dass die ursprünglichen Daten unverändert bleiben.
3. Preisanpassung, wenn verfügbar: Wenn der Produktname in den neuen Preisen gefunden wird, wird der Preis in der Produktkopie aktualisiert.
4. Hinzufügen des aktualisierten Produkts zur neuen Liste: Die aktualisierte Produktkopie wird zur neuen Liste von Produkten hinzugefügt.
5. Rückgabe der aktualisierten Produktliste: Nachdem alle Produkte durchlaufen wurden, wird die neue aktualisierte Produktliste zurückgegeben.

Gesamtpreis berechnen:

```
def calculate_total(products):
    total = 0
    for product in products:
        total += product['price'] * product['quantity']
    return total
```

Vorgehen

Die Funktion zur Berechnung des Gesamtpreises zielt darauf ab, den Gesamtpreis der Produkte in der Einkaufsliste basierend auf den Preisen und Mengen zu berechnen.

1. Initialisierung des Gesamtpreises: Eine Variable wird verwendet, um den Gesamtpreis zu speichern, und sie wird mit 0 initialisiert.
2. Durchlaufen der Produkte in der Liste: Für jedes Produkt in der Liste wird der Preis mit der Menge multipliziert, um den Gesamtpreis für dieses spezielle Produkt zu berechnen.
3. Addition des Produktgesamtpreises zum Gesamtpreis: Der berechnete Gesamtpreis für das Produkt wird zum laufenden Gesamtpreis addiert.
4. Rückgabe des Gesamtpreises: Nachdem alle Produkte durchlaufen wurden, wird der endgültige Gesamtpreis zurückgegeben.

Testen Sie Ihre Funktionen:

```
if __name__ == '__main__':
    updated_products = update_prices(products, prices)
    print("Ursprüngliche Produkte:", products)
    print("Aktualisierte Produkte:", updated_products)
    print("Gesamtpreis (alt):", calculate_total(products))
    print("Gesamtpreis (neu):", calculate_total(updated_products))
```

Zusammenfassung

```
products = [
    {'product': 'Apple', 'quantity': 5, 'price': 1.2},
    {'product': 'Banana', 'quantity': 2, 'price': 0.8},
    # Add more products as needed
]

new_prices = {
    'Apple': 0.5,
    'Banana': 0.3,
    # Add more prices as needed
}


def update_prices(products, new_prices):
    updated_products = []
    for product in products:
        updated_product = product.copy()
        product_name = updated_product['product']
        if product_name in new_prices:
            updated_product['price'] = new_prices[product_name]
        updated_products.append(updated_product)
    return updated_products


def calculate_total(products):
    total = 0
```

```
    for product in products:
        total += product['price'] * product['quantity']
    return total


if __name__ == '__main__':
    updated_products = update_prices(products, prices)
    print("Ursprüngliche Produkte:", products)
    print("Aktualisierte Produkte:", updated_products)
    print("Gesamtpreis (alt):", calculate_total(products))
    print("Gesamtpreis (neu):", calculate_total(updated_products))
```
