# LU07.A06 - Unpure wird pure

Entfernen Sie den globalen Zustand aus einer kleinen Lagerverwaltung und trennen Sie Berechnung von Ein- und Ausgabe.

## Ausgangslage

```
lager = {}
protokoll = []


def einlagern(artikel, menge):
    global lager
    if artikel in lager:
        lager[artikel] += menge
    else:
        lager[artikel] = menge
    protokoll.append(f"+{menge} {artikel}")
    print(f"{artikel}: {lager[artikel]}")


def entnehmen(artikel, menge):
    global lager
    if lager.get(artikel, 0) < menge:
        print("zu wenig Bestand")
        return False
    lager[artikel] -= menge
    protokoll.append(f"-{menge} {artikel}")
    return True
```

```
einlagern("maus", 5)
einlagern("maus", 3)
entnehmen("maus", 2)      # True
entnehmen("maus", 99)     # False
print(lager)              # {'maus': 6}
```

## Detaillierte Aufgabenstellung

1. Schreiben Sie einen Test, der die obige Abfolge und ihr Endergebnis festhält.
2. Bauen Sie einen **funktionalen Kern**: Funktionen, die den Bestand als Argument bekommen und einen **neuen** Bestand zurückgeben, ohne ihn zu mutieren und ohne `print`.
3. Bauen Sie eine dünne **Schale**, in der Ausgabe und das Fortschreiben des Zustands stattfinden.
4. Nach dem Umbau darf im Kern kein `global`, kein `print` und keine Mutation der Argumente vorkommen.
5. Belegen Sie mit Tests, dass der Kern pure ist: Zweimal derselbe Aufruf mit demselben Bestand

liefert dasselbe Ergebnis, und der übergebene Bestand ist danach unverändert.

## Leitfragen

- Wie gibt eine Funktion gleichzeitig den neuen Bestand und die Information «hat geklappt / hat nicht geklappt» zurück?
- Wohin gehört das Protokoll - in den Kern oder in die Schale?
- Was wird an dieser Version einfacher zu testen als vorher?

---

M323-LU07, M323-D1I

© Kevin Maurizi

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
**https://wiki.bzz.ch/modul/m323/learningunits/lu07/aufgaben/purerefactoring?rev=1788945298**

Last update: **2026/09/09 11:14**