

(Die nachfolgenden Aufgaben wurden mit Hilfe von ChatGPT erstellt, überprüft und ggf. angepasst)

Übungsaufgaben zur 3. Vorlesung: positional und key-word Argumente, mutable und immutable Variablen, map, lambda, dictionaries

1. Aufgabenstellung: Statistikbericht mit Optionen

Schreibe eine Funktion `erstelle_bericht(*werte, **optionen)`, die: Eine beliebige Anzahl von Zahlen (`*werte`) als Argumente akzeptiert. Optional über `**optionen` steuert, welche statistischen Werte berechnet und ausgegeben werden sollen. Die verfügbaren Optionen sollen sein:

"mittelwert" → gibt den Mittelwert der Zahlen aus

"max" → gibt den größten Wert aus

"min" → gibt den kleinsten Wert aus

"anzahl" → gibt die Anzahl der übergebenen Werte aus

Die Funktion soll am Ende für jede gesetzte Option den entsprechenden Wert ausgeben.

Beispielaufruf:

```
erstelle_bericht(3, 5, 7, 2, 9, mittelwert=True, max=True, anzahl=True)
```

Ausgabe:

Mittelwert: 5.2

Max: 9

Anzahl: 5

Hinweise:

Verwende `*werte`, um eine variable Anzahl von Zahlen entgegenzunehmen.

Verwende `**optionen`, um optionale Ausgaben über Schlüsselwörter zu steuern.

Nutze `sum()`, `len()`, `max()`, `min()` zur Berechnung.

Gib die Werte nur aus, wenn die jeweilige Option in `optionen` enthalten und `True` ist.

Zusatzaufgaben:

Gib eine Warnung aus, wenn keine Zahlen übergeben wurden.

Runde den Mittelwert auf 2 Nachkommastellen.

2. Aufgabenstellung: Mutable und Immutable Variables

Betrachte folgenden Code

```
def verarbeite_liste(liste):  
    liste.append(100)  
    liste = [0, 0, 0]  
  
def verarbeite_zahl(x):  
    x += 1  
  
zahlen = [1, 2, 3]  
wert = 10  
  
verarbeite_liste(zahlen)  
verarbeite_zahl(wert)  
  
print("zahlen:", zahlen)  
print("wert:", wert)
```

- Was wird durch die print-Befehle ausgegeben?
- Erkläre warum sich das Verhalten der beiden Funktionen unterscheidet.
- Wie könnte man „copy“ benutzen, um Änderungen der Eingabe in „verarbeite_liste“ zu vermeiden?
- Wie könnte man „verarbeite_zahl“ ändern, wenn man möchte, dass sich der Eingabewert nach Aufruf der Funktion auch außerhalb der Funktion ändert?

3. Aufgabenstellung: Preisliste mit Mehrwertsteuer berechnen

Du hast eine Liste von Nettopreisen und möchtest daraus die **Bruttopreise** berechnen – also inklusive **19 % Mehrwertsteuer**.

Gegeben:

nettopreise = [10.0, 23.5, 7.99, 100.0]

Aufgabenstellung:

Verwende map() und eine lambda-Funktion, um eine neue Liste bruttopreise zu erstellen, in der jeder Preis um 19 % erhöht wurde.

4. Aufgabenstellung: Verarbeitung einer Produktliste (Anspruchsvolle Aufgabe mit map und lambda)

Gegeben ist eine Liste von Produkten, wobei jedes Produkt ein Tupel mit folgendem Aufbau ist:

(name, nettopreis, steuersatz_in_prozent)

```
produkte = [  
    ("Buch", 12.0, 7),  
    ("Laptop", 999.0, 19),  
    ("Wasserflasche", 1.5, 19),  
    ("Apfel", 0.80, 7)  
]
```

Aufgabenstellung:

1. Verwende map() und eine lambda-Funktion, um aus der obigen Liste eine neue Liste bruttoprodukte zu erzeugen.
Jeder Eintrag soll ein **neues Tupel** sein: (name, bruttopreis), wobei der Bruttopreis berechnet wird durch:
$$\text{bruttopreis} = \text{nettopreis} \times (1 + \text{steuersatz} / 100)$$
2. Gib die Bruttopreise **auf zwei Nachkommastellen gerundet** aus, indem du die Funktion „round“ verwendest

Beispielausgabe:

```
[('Buch', 12.84), ('Laptop', 1188.81), ('Wasserflasche', 1.79), ('Apfel', 0.86)]
```

5. Aufgabenstellung: Würfelspiel – Punkte zählen mit einem Dictionary

In einem einfachen Würfelspiel spielen mehrere Personen gegeneinander. Jede Person würfelt **dreimal**, und ihre Ergebnisse werden gespeichert.

Du bekommst ein Dictionary, in dem die Spielernamen als **Schlüssel (keys)** und ihre drei Würfelresultate (als Liste) als **Werte (values)** gespeichert sind:

```
ergebnisse = {  
    "Alice": [4, 5, 6],  
    "Bob": [3, 2, 6],  
    "Charlie": [1, 1, 1],  
    "Dana": [6, 6, 6]  
}
```

Aufgabenstellung:

- Schreibe ein Programm, das für jeden Spieler die **Gesamtpunktzahl** (Summe der drei Würfe) berechnet.
- Gib für jeden Spieler folgendes aus:
"Alice hat 15 Punkte."
- Ermittle anschließend, **wer gewonnen hat** (d. h. den höchsten Punktwert erreicht hat).
- Gib den Gewinner im Format aus:
"Dana hat gewonnen mit 18 Punkten!"

Bonus (optional):

- Bei Gleichstand sollen **alle Gewinner** ausgegeben werden:
"Bob und Dana haben gewonnen mit 17 Punkten!"

Hinweise:

- Du kannst `sum()` verwenden, um die Punktzahl zu berechnen.
- Verwende eine Schleife über `ergebnisse.items()` für die Verarbeitung.
- Den/die Gewinner kannst du z. B. durch Vergleich mit dem aktuellen Maximalwert finden.

6. Aufgabenstellung: Punkteausswertung im Kartenspiel (Sehr anspruchsvolle Aufgabe mit Dictionaries und der Logik eines Programms)

In einem Kartenspiel sammeln mehrere Spieler Karten mit unterschiedlichen Farben und Punktwerten. Jede Karte hat eine **Farbe** ("rot", "blau", "grün", "gelb") und einen **Wert** (zwischen 1 und 10).

Die Datenstruktur ist ein **verschachteltes Dictionary** mit folgendem Aufbau:

```
karten = {  
    "Alice": [("rot", 5), ("blau", 3), ("rot", 4)],  
    "Bob":  [("grün", 10), ("rot", 2)],  
    "Clara": [("blau", 1), ("gelb", 9), ("gelb", 3)],  
    "David": []  
}
```

Aufgabenstellung:

- Berechne für jeden Spieler die **Gesamtpunktzahl**.
- Gib zusätzlich die **Punkte nach Farben getrennt** aus.
Beispiel: "Alice: gesamt 12 Punkte (rot: 9, blau: 3)"
Verwende für die Farbpunkte ein zweites Dictionary (z. B. farbe_zu_punkten = {}).
- Ermittle den Spieler mit den **meisten Punkten** (wie im vorherigen Beispiel).
- Berücksichtige auch den Fall, dass ein Spieler **keine Karten** hat (→ 0 Punkte).

Beispielausgabe:

Alice: gesamt 12 Punkte (rot: 9, blau: 3)

Bob: gesamt 12 Punkte (grün: 10, rot: 2)

Clara: gesamt 13 Punkte (blau: 1, gelb: 12)

David: gesamt 0 Punkte

Clara hat gewonnen mit 13 Punkten!