

Brick-Flow staffelt den Palettenpreis: unter 10 Paletten **80 €**, ab 10 **76 €**, ab 50 **72 €** — jeweils für die *gesamte* Bestellung.

```
def preis_je_palette(menge):  
    if menge >= 50:  
        return 72  
    elif menge >= 10:  
        return 76  
    else:  
        return 80  
  
def warenwert(menge):  
    preis = preis_je_palette(menge)  
    return menge * preis  
  
print(warenwert(12))    # 912
```



Aktivierung — Jede/r für sich: (a) Berechnen

Sie `warenwert(49)` und `warenwert(50)`.

(b) Was fällt auf — und was würden Sie einem Kunden raten, der 48 Paletten bestellen will?

Brick-Flow kauft Granulat ein (*Bezugskalkulation*). Zwei Angebote liegen vor. **Skonto** gibt es nur bei Zahlung **innerhalb von 10 Tagen**.

	Listenpreis	Liefererrabatt	Skonto	Bezugskosten (Fracht)
Lieferant A	2000 €	10 %	2 %	60 €
Lieferant B	1870 €	—	4 %	frei Haus

Rechenweg: Listenpreis – Rabatt = Zieleinkaufspreis; – Skonto (falls rechtzeitig) = Bareinkaufspreis; + Bezugskosten = **Bezugspreis**.

- a) `bezugspreis(listenpreis, rabatt, skonto, bezugskosten, zahlung_nach_tagen)`
- b) `guenstiger(preis_a, preis_b)` gibt "A", "B" oder "gleich" zurück
- c) Welcher Lieferant gewinnt bei Zahlung nach **7** Tagen, welcher nach **30**?



Aktivierung — Zu zweit: Lösen Sie a)–c) im Notebook. Kontrollwert:
Lieferant A, Zahlung nach 7 Tagen → **1824.0**.

Ein Außendienstmitarbeiter erhält monatlich:

- ein **Fixum** von **2500 €**,
- **2 %** Provision auf den Umsatz **bis 40.000 €**,
- **4 %** Provision *nur auf den Teil* des Umsatzes **über 40.000 €**,
- einen **Zielbonus von 1000 €**, wenn das Umsatzziel erreicht wird (Standard: **60.000 €**).

- a) `provision(umsatz)` — nur die Provision
- b) `monatsgehalt(umsatz, ziel=60000)` — nutzt `provision`
- c) Testen Sie mit **30.000**, **50.000** und **70.000 €** Umsatz. Was ändert sich bei 70.000 €, wenn das Ziel auf 80.000 € steigt?



Aktivierung — Zu zweit: Rechnen Sie `provision(70000)` zuerst *von Hand* — erst dann in Python. Kontrollwert: `provision(30000) → 600.0`.

Ein Bauteil kann Brick-Flow selbst fertigen (**12.000 €** Fixkosten + **18 €** je Stück) oder zukaufen (**26 €** je Stück, **ab 2000 Stück 22 €**). Ein Kollege hat schon begonnen:

```
def kosten_eigen(menge):  
    return 12000 + 18 * menge  
  
def kosten_fremd(menge):  
    if menge >= 2000:  
        return 22 * menge  
    else:  
        26 * menge
```

- a) `kosten_eigen(1000) - kosten_fremd(1000)` stürzt ab, mit 2400 nicht. Warum? Reparieren Sie.
- b) `entscheidung(menge)` gibt "Eigenfertigung", "Fremdbezug" oder "gleich" zurück.
- c) Testen Sie 1000, 1800, 2400 und 4000 Stück — und erklären Sie das Ergebnis.

Was Sie heute geübt haben

- Mehrstufige Entscheidungen (**if/elif/else**) in Funktionen
- Funktionen, die andere Funktionen nutzen — jede Rechnung steht nur einmal im Code
- Typische BWL-Rechenwege: Bezugskalkulation, Stufenprovision, Make-or-Buy
- Fehlendes **return**: Der Fehler zeigt sich erst bei bestimmten Eingaben



Aktivierung — Zum Mitnehmen: Testen Sie eine Funktion immer mit Werten *auf beiden Seiten* jeder Grenze — und genau *auf der* Grenze (z. B. 1999, 2000, 2001).