DS 9 – Weitere Übungen

Übung: Funktionen lesen, testen und erweitern

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Die Aufgaben stehen auch im Skript im Abschnitt „Weitere Übungen“; die Lösungen im Lösungsheft.

Aufgabe 1: Verschachtelte Aufrufe ●○○

Lies das Programm, **ohne** es auszuführen. Gib an, was jede `print`-Zeile ausgibt. Überprüfe danach am Computer.

```python
def plus_eins(x):
    return x + 1

def doppelt(x):
    return 2 * x

print(doppelt(plus_eins(3)))
print(plus_eins(doppelt(3)))
print(doppelt(doppelt(plus_eins(0))))
```

```
8
7
4
```

Deine Antwort:

…

Aufgabe 2: Kreisfläche ●○○

Die Fläche eines Kreises berechnet man mit der Formel $A = \pi \cdot r \cdot r$.

1. Implementiere eine Funktion `kreisflaeche(radius)`, die die Fläche zurückgibt. Verwende `math.pi`.
2. Teste die Funktion mit den Radien 1, 0 und 2. Zur Kontrolle: Für den Radius 1 muss genau π herauskommen, also etwa 3,14.

Aufgabe 3: Gute Testwerte wählen ●○○

Die Funktion `ist_teenager(alter)` soll `True` zurückgeben, wenn jemand 13 bis einschließlich 19 Jahre alt ist, sonst `False`.

```python
def ist_teenager(alter):
    return alter >= 13 and alter <= 19
```

Nenne vier Testwerte, mit denen du die Funktion besonders gut prüfen kannst, und gib jeweils das erwartete Ergebnis an. Begründe deine Auswahl in einem Satz.

Deine Antwort:

…

Aufgabe 4: Ungerade mit Hilfsfunktion ●○○

Im Abschnitt über Funktionen hast du `ist_gerade(zahl)` geschrieben:

```python
def ist_gerade(zahl):
    return zahl % 2 == 0
```

Implementiere eine Funktion `ist_ungerade(zahl)`, die `True` zurückgibt, wenn die Zahl ungerade ist. Benutze dafür die Funktion `ist_gerade` und den Operator `not`. Teste mit 7 und mit 10.

Aufgabe 5: Wo steckt der Fehler? ●●○

Eine Fahrt dauert 2 Stunden und 15 Minuten. Das Programm soll die Fahrzeit in Minuten berechnen, also 135. Es gibt aber 215 aus.

```python
def stunden_zu_minuten(stunden):
    return stunden * 100

def fahrzeit_minuten(stunden, minuten):
    return stunden_zu_minuten(stunden) + minuten

print(fahrzeit_minuten(2, 15))
```

```
215
```

1. Beschreibe, wie du durch einen **einzelnen Test der Hilfsfunktion** herausfindest, in welcher der beiden Funktionen der Fehler steckt.
2. Behebe den Fehler und teste erneut.

Deine Antwort:

…

Aufgabe 6: Versandkosten ●●○

Ein Online-Shop berechnet 5 Euro Versandkosten. Ab einem Warenwert von 30 Euro ist der Versand kostenlos.

1. Implementiere eine Funktion `versandkosten(warenwert)`, die die Versandkosten zurückgibt (0 oder 5).
2. Implementiere eine Funktion `endpreis(warenwert)`, die den Warenwert plus Versandkosten zurückgibt. Sie soll `versandkosten` aufrufen.
3. Teste `endpreis` mit den Warenwerten 20, 30 und 45. Überlege vorher, was herauskommen muss.

Aufgabe 7: Pizza-Vergleich ●●○

Was ist mehr Pizza: **eine** große Pizza mit 32 cm Durchmesser oder **zwei** kleine mit je 22 cm Durchmesser?

1. Implementiere eine Funktion `pizza_flaeche(durchmesser)`, die die Fläche einer runden Pizza zurückgibt. Sie soll die Funktion `kreisflaeche(radius)` aus dem Auftrag *Kreisfläche* aufrufen. Denk daran: Der Radius ist der halbe Durchmesser.
2. Berechne damit die Fläche der großen Pizza und die Fläche der beiden kleinen Pizzen zusammen. Gib beide Werte gerundet aus und beantworte die Frage.

Aufgabe 8: Primzahlen mit Hilfsfunktionen ●●●

Im Auftrag *Primzahl-Test* (Übung zu for- und while-Schleife) hast du geprüft, ob eine Zahl eine Primzahl ist. Jetzt zerlegst du die Aufgabe in Funktionen.

1. Implementiere eine Funktion `ist_teiler(t, zahl)`, die `True` zurückgibt, wenn `t` ein Teiler von `zahl` ist (die Division geht ohne Rest auf).
2. Implementiere eine Funktion `ist_primzahl(zahl)`, die `True` oder `False` zurückgibt. Sie soll `ist_teiler` benutzen.

3. Gib mit einer Schleife alle Primzahlen von 1 bis 30 aus.

> **Tipp:** Eine Primzahl ist eine Zahl größer als 1, die nur durch 1 und sich selbst teilbar ist. Gehe in `ist_primzahl` wie im Auftrag *Primzahl-Test* vor: Starte mit `ergebnis = zahl > 1` und prüfe in einer Schleife alle möglichen Teiler von 2 bis `zahl - 1`. Findest du einen Teiler, setzt du `ergebnis = False`.