DS 8 – Weitere Übungen

Funktionen: Programme aus Bausteinen

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: Fachbegriffe am Beispiel ●○○

Am Kiosk kostet jede Pfandflasche 25 Cent Pfand zusätzlich zum Preis. Lies das Programm:

```
def preis_mit_pfand(preis, flaschen):
    return preis + flaschen * 0.25

gesamt = preis_mit_pfand(3.5, 6)
print(gesamt)
```

```
5.0
```

Nenne zu diesem Programm:

1. den Namen der Funktion,
2. die Parameter,
3. die Argumente beim Aufruf,
4. die Zeile, in der die Funktion aufgerufen wird,
5. den Rückgabewert für diesen Aufruf.

Deine Antwort:

…

Aufgabe 2: Rückgabewerte vorhersagen ●○○

Gegeben ist die Funktion `rechne`:

```
def rechne(x):
    return 3 * x - 1
```

Bestimme **ohne Computer**, welchen Wert die Aufrufe zurückgeben. Prüfe danach am Computer, z. B. mit `print(rechne(2))`.

Aufruf	Rückgabewert
`rechne(2)`	
`rechne(0)`	
`rechne(-1)`	
`rechne(10)`	
`rechne(rechne(1))`	

Aufgabe 3: Eintritt im Freibad ●○○

Im Freibad kostet der Eintritt 4 Euro. Für jede Stunde, die man bleibt, kommen 2 Euro dazu.

1. Implementiere eine Funktion `eintritt(stunden)`, die den Preis in Euro **zurückgibt**.

2. Teste sie mit den Werten aus der Tabelle. Rechne die erwarteten Werte vorher von Hand aus.

Aufruf	erwartet	tatsächlich
`eintritt(1)`		
`eintritt(3)`		
`eintritt(0)`		

Aufgabe 4: Drei Fehler ●○○

Die Funktion soll die Fläche eines Dreiecks berechnen (Grundseite mal Höhe durch 2). Das Programm enthält **drei Fehler**. Finde sie und schreibe das Programm richtig auf. Teste es: Für die Grundseite 4 und die Höhe 3 muss `6.0` herauskommen.

```
def flaeche_dreieck(g, h)
return g * h / 2

print(flaeche_dreieck(4))
```

```
  File "<Zelle 3>", line 1
    def flaeche_dreieck(g, h)
                              ^
SyntaxError: expected ':'
```

Deine Antwort:

…

Aufgabe 5: Ein Test deckt den Fehler auf ●●○

Die Funktion soll den Mittelwert von zwei Zahlen zurückgeben. Sie läuft ohne Fehlermeldung.

```
def mittelwert(a, b):
    return a + b / 2
```

1. Rechne von Hand aus, was für `mittelwert(4, 6)` herauskommen **muss**.
2. Teste die Funktion mit `mittelwert(4, 6)` und mit `mittelwert(0, 0)`. Notiere die Ergebnisse.
3. Erkläre den Fehler und verbessere die Funktion.
4. Erkläre, warum der Test mit `0` und `0` allein den Fehler nicht gefunden hätte.

Deine Antwort:

…

Aufgabe 6: Kinokarten ●●○

Ein Kino verlangt je nach Alter einen anderen Preis:

- unter 6 Jahren: 0 Euro (frei),
- von 6 bis unter 18 Jahren: 6 Euro,
- ab 18 Jahren: 9 Euro.

1. Implementiere eine Funktion `kinopreis(alter)`, die den Preis **zurückgibt**. Verwende eine mehrseitige Verzweigung.
2. Teste die Funktion an den Grenzen: mit den Altern 5, 6, 17 und 18.

Aufgabe 7: Funktion in der Schleife ●●○

Fülle das Ablaufprotokoll des Hauptprogramms aus. Trage nach jedem Schritt die Werte von `i` und `summe` ein (– bedeutet: noch kein Wert). Gib dann an, was das Programm ausgibt.

```
def dreifach(zahl):
    return 3 * zahl

summe = 0
for i in range(1, 4):
    summe = summe + dreifach(i)
print("Summe:", summe)
```

```
Summe: 18
```

Schritt	Zeile	Anweisung	i	summe
1	1	`def dreifach(zahl):`	–	–
2	4	`summe = 0`	–	
3	5	`for i in range(1, 4):`		
4	6	`summe = summe + dreifach(i)`		
5	5	`for i in range(1, 4):`		
6	6	`summe = summe + dreifach(i)`		
7	5	`for i in range(1, 4):`		
8	6	`summe = summe + dreifach(i)`		
9	5	`for i in range(1, 4):`		
10	7	`print("Summe:", summe)`		

Deine Antwort:

…

Aufgabe 8: Quersumme als Funktion ●●●

Im Auftrag *Quersumme* (Abschnitt über die while-Schleife) hast du die Quersumme einer Zahl berechnet (Quersumme von 384 ist 3 + 8 + 4 = 15). Zur Erinnerung: `zahl % 10` liefert die letzte Ziffer, `zahl // 10` entfernt sie.

1. Implementiere eine Funktion `quersumme(zahl)`, die die Quersumme **zurückgibt**. Teste sie mit 384 (erwartet: 15), 7 (erwartet: 7) und 1000 (erwartet: 1).
2. Benutze die Funktion in einer `for`-Schleife: Gib alle Zahlen von 1 bis 100 aus, deren Quersumme genau 10 ist.

> **Tipp:** Schreibe die Schleife aus dem Abschnitt über die while-Schleife in den Funktionsrumpf und ersetze das `print` am Ende durch `return`. Im zweiten Teil steht der Aufruf `quersumme(i)` in der Bedingung einer `if`-Anweisung, so wie `ist_gerade(i)` im Auftrag *Gerade Zahlen*.