DS 7 – Weitere Übungen

Übung: for- oder while-Schleife?

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: Welche Schleife passt? ●○○

Entscheide jeweils, ob eine for- oder eine while-Schleife besser passt, und begründe kurz. Du musst die Programme nicht schreiben.

Situation	for oder while?
a) das Taschengeld von 12 Monaten zusammenzählen	
b) Zahlen einlesen, bis eine negative Zahl eingegeben wird	
c) alle geraden Zahlen von 2 bis 50 ausgeben	
d) so lange nachfragen, bis `ja` oder `nein` eingegeben wird	
e) eine eingegebene Zahl so oft halbieren, bis sie kleiner als 1 ist	
f) für jede der 25 Personen einer Klasse eine Note einlesen	

Deine Antwort:

…

Aufgabe 2: Aus for wird while ●○○

Schreibe ein Programm mit einer **while**-Schleife, das genau dasselbe ausgibt wie das folgende Programm. Denke an Startwert, Bedingung und Veränderung der Zählvariablen.

```
for i in range(5, 26, 5):
    print(i)
```

```
5
10
15
20
25
```

Aufgabe 3: Nur die ungeraden ●○○

Lies das Programm, **ohne** es auszuführen.

1. Gib an, was das Programm ausgibt.
2. Gib an, wie oft die Zeile `summe = summe + zahl` ausgeführt wird.

```
summe = 0
for zahl in range(1, 6):
    if zahl % 2 == 1:
        summe = summe + zahl
print(summe)
```

```
9
```

Deine Antwort:

…

Aufgabe 4: Potenzieren ohne ** ●○○

Das Programm soll eine Potenz berechnen, ohne den Operator `**` zu benutzen. Dafür wird die Basis so oft mit sich selbst multipliziert, wie der Exponent angibt: $2^5 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2$. Ergänze die drei Lücken `___`. Teste mit Basis `2` und Exponent `10` (Ergebnis 1024) und mit Basis `5` und Exponent `0` (Ergebnis 1).

Lückencode

```
basis = int(input("Basis: "))
exponent = int(input("Exponent: "))
ergebnis = ___
for i in range(___):
    ergebnis = ___
print(basis, "hoch", exponent, "ist", ergebnis)
```

Aufgabe 5: Fehler finden: Vielfache von 7 zählen ●●○

Das Programm soll zählen, wie viele Zahlen von 1 bis 70 durch 7 teilbar sind. Richtig wäre `Anzahl: 10`. Es enthält **drei Fehler**.

```
for zahl in range(1, 70):
    if zahl % 7 = 0:
        anzahl = anzahl + 1
print("Anzahl:", anzahl)
```

```
  File "<Zelle 4>", line 2
    if zahl % 7 = 0:
       ^^^^^^^^
SyntaxError: cannot assign to expression here. Maybe you meant '==' instead of '='?
```

1. Finde die drei Fehler. Notiere jeweils die Zeile und was falsch ist.
2. Schreibe das Programm richtig auf.

Deine Antwort:

…

Aufgabe 6: Alle Teiler einer Zahl ●●○

Schreibe ein Programm, das eine ganze Zahl einliest und alle ihre Teiler ausgibt. Am Ende soll es ausgeben, wie viele Teiler es sind. Begründe, welche Schleifenart du verwendest.

So könnte es aussehen:

Ausgabe:

```
Zahl: «12»
1
2
3
4
6
12
Anzahl der Teiler: 6
```

Deine Antwort:

…

Aufgabe 7: Gerade und ungerade zählen ●●○

Schreibe ein Programm, das so lange ganze Zahlen einliest, bis `0` eingegeben wird. Danach soll es ausgeben, wie viele der eingegebenen Zahlen gerade und wie viele ungerade waren. Die `0` zählt nicht mit. Teste mit `4`, `7`, `10`, `3`, `8`, `0` und mit sofort `0`.

Aufgabe 8: Bestweite beim Weitsprung ●●●

Beim Weitsprung werden die Weiten eines Versuchs nach dem anderen eingegeben (in Metern, als Kommazahl). Die Eingabe `0` beendet die Eingabe. Schreibe ein Programm, das danach die **Anzahl der Versuche** und die **beste Weite** ausgibt. Wurde gar keine Weite eingegeben, soll es „Keine Weite eingegeben.“ ausgeben.

So könnte es aussehen:

Ausgabe:

```
Weite in m (0 = Ende): «3.85»
Weite in m (0 = Ende): «4.10»
Weite in m (0 = Ende): «3.92»
Weite in m (0 = Ende): «0»
Versuche: 3
Bestweite: 4.1 m
```

> **Tipp:** Merke dir die bisher beste Weite in einer Variablen `beste`. Weil alle Weiten größer als 0 sind, kann sie bei 0 starten. Bei jeder neuen Weite fragst du: Ist sie größer als die bisher beste? Dann wird sie die neue beste Weite.