DS 6 – Weitere Übungen

Die while-Schleife: Wiederholen, solange eine Bedingung gilt

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: In Viererschritten ●○○

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

1. Schreibe auf, was es ausgibt.
2. Gib an, wie oft der Schleifenrumpf ausgeführt wird und welchen Wert `zahl` am Ende hat.

```
1 zahl = 20
2 while zahl > 5:
3     print(zahl)
4     zahl = zahl - 4
5 print("fertig")
```

```
20
16
12
8
fertig
```

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

Ausgabe:

```
20
16
12
8
fertig
```

Lösung

Der Rumpf wird **viermal** ausgeführt. Danach hat `zahl` den Wert `4`. Die Bedingung `4 > 5` ist falsch, die Schleife endet. Die `4` wird nicht mehr ausgegeben, denn die Bedingung wird **vor** dem Rumpf geprüft. Wer `fertig` vergessen hat: Zeile 5 ist nicht eingerückt und wird nach der Schleife einmal ausgeführt.

Aufgabe 2: Countdown ●○○

Das Programm soll eine Startzahl einlesen und von dort bis 1 herunterzählen. Danach soll es „Start!“ ausgeben. Ergänze die beiden Lücken `___` und teste mit `5` und mit `0`.

Lückencode

```
zahl = int(input("Starte bei: "))
while ___:
    print(zahl)
    ___
print("Start!")
```

Lösung

✅ **Lösung**

Lösung

```
zahl = int(input("Starte bei: "))
while zahl > 0:
    print(zahl)
    zahl = zahl - 1
print("Start!")
```

```
Starte bei: 5
5
4
3
2
1
Start!
```

Lösung

Getestet mit den Eingaben: `5`

Lösung

Ausgabe:

```
Starte bei: «5»
5
4
3
2
1
Start!
```

Lösung

Die Zahl muss im Rumpf **kleiner** werden, sonst bleibt die Bedingung immer wahr. Auch `zahl >= 1` ist als Bedingung richtig. Bei der Eingabe `0` ist die Bedingung schon beim ersten Prüfen falsch: Der Rumpf wird nie ausgeführt, und es erscheint nur `Start!`.

Aufgabe 3: Anzahl der Spieler ●○○

Ein Brettspiel ist für 2 bis 4 Personen. Schreibe ein Programm, das die Anzahl der Spielerinnen und Spieler so lange abfragt, bis eine gültige Zahl eingegeben wird. Bei einer ungültigen Eingabe soll es einen Hinweis ausgeben. Am Ende soll es `Das Spiel startet mit 3 Personen.` ausgeben (mit der eingegebenen Zahl).

Lösung

✅ **Lösung**

Lösung

```
anzahl = int(input("Wie viele spielen mit? "))
while anzahl < 2 or anzahl > 4:
    print("Das Spiel ist für 2 bis 4 Personen.")
    anzahl = int(input("Wie viele spielen mit? "))
print("Das Spiel startet mit", anzahl, "Personen.")
```

```
Wie viele spielen mit? 4
Das Spiel startet mit 4 Personen.
```

Lösung

Getestet mit den Eingaben: `6`, `1`, `3`

Lösung

Ausgabe:

```
Wie viele spielen mit? «6»
Das Spiel ist für 2 bis 4 Personen.
Wie viele spielen mit? «1»
Das Spiel ist für 2 bis 4 Personen.
Wie viele spielen mit? «3»
Das Spiel startet mit 3 Personen.
```

Lösung

Die Bedingung beschreibt, wann **noch einmal** gefragt werden muss: bei weniger als 2 **oder** mehr als 4 Personen. Mit `and` wäre die Bedingung nie wahr. Teste auch die Grenzwerte `2` und `4` (gültig) sowie `1` und `5` (ungültig).

Aufgabe 4: Endlosschleife oder nicht? ●●○

Entscheide für jedes Programm, ob die Schleife irgendwann endet oder ob eine Endlosschleife entsteht. Begründe kurz. Führe die Programme **nicht** aus.

a)

```
i = 1
while i < 5:
    i = i + 1
```

b)

```
i = 10
while i > 0:
    i = i + 1
```

Nach 5 Sekunden abgebrochen: Die Schleife hört nicht auf (Endlosschleife). In Jupyter stoppst du sie mit ■ (Interrupt).

c)

```
i = 1
while i != 10:
    i = i + 2
```

Nach 5 Sekunden abgebrochen: Die Schleife hört nicht auf (Endlosschleife). In Jupyter stoppst du sie mit ■ (Interrupt).

d)

```
i = 0
while i < 3:
    print(i)
i = i + 1
```

```
0
0
0
0
0
0
0
0
0
0
0
0
…
```

Hier abgebrochen: Die Ausgabe hört nicht auf (Endlosschleife). In Jupyter stoppst du sie mit ■ (Interrupt).

Deine Antwort:

…

Lösung

✅ Lösung

Lösung

- a) **Endet.** `i` wird größer und erreicht nach vier Durchläufen den Wert 5. Dann ist `5 < 5` falsch.
- b) **Endlosschleife.** `i` wird größer statt kleiner. Damit bleibt `i > 0` immer wahr.
- c) **Endlosschleife.** `i` nimmt die Werte 1, 3, 5, 7, 9, 11, … an. Die 10 wird übersprungen, deshalb ist `i != 10` immer wahr. Mit der Bedingung `i < 10` würde die Schleife enden.
- d) **Endlosschleife.** `i = i + 1` ist nicht eingerückt und gehört nicht zum Rumpf. Im Rumpf steht nur `print(i)`, und es wird endlos `0` ausgegeben.

Fall c) zeigt: Eine Bedingung mit `!=` ist gefährlich, wenn die Variable in größeren Schritten springt. Sie kann den Wert, bei dem die Schleife enden soll, „überspringen“.

Aufgabe 5: Fehler finden: Wörter zählen ●●○

Das Programm soll so lange Wörter einlesen, bis `stopp` eingegeben wird. Danach soll es ausgeben, wie viele Wörter eingegeben wurden (ohne `stopp`). Es enthält **drei Fehler**.

```
anzahl = 0
wort = input("Wort (stopp zum Beenden): ")
while wort == "stopp":
    anzahl = anzahl + 1
wort = input("Wort (stopp zum Beenden): ")
print("Du hast", anzahl "Wörter eingegeben.")
```

```
  File "<Zelle 9>", line 6
    print("Du hast", anzahl "Wörter eingegeben.")
                            ^^^^^^^^^^^^^^^^^^^^
SyntaxError: invalid syntax
```

1. Finde die drei Fehler. Notiere jeweils die Zeile und was falsch ist.
2. Schreibe das Programm richtig auf. Teste es mit den Eingaben `Hund`, `Katze`, `Maus`, `stopp` und auch mit sofort `stopp`.

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

1. **Zeile 3:** Die Bedingung ist umgekehrt. Die Schleife soll laufen, **solange** das Wort **nicht** `stopp` ist: `wort != "stopp"`.
2. **Zeile 5:** Die nächste Eingabe ist nicht eingerückt und gehört nicht zum Rumpf. Dann ändert sich `wort` in der Schleife nie, und es entsteht eine Endlosschleife.
3. **Zeile 6:** Zwischen `anzahl` und dem Text fehlt ein Komma (`SyntaxError`).

Lösung

```
anzahl = 0
wort = input("Wort (stopp zum Beenden): ")
while wort != "stopp":
    anzahl = anzahl + 1
    wort = input("Wort (stopp zum Beenden): ")
print("Du hast", anzahl, "Wörter eingegeben.")
```

```
Wort (stopp zum Beenden): Informatik
Wort (stopp zum Beenden): Python
Wort (stopp zum Beenden): stopp
Du hast 2 Wörter eingegeben.
```

Lösung

Getestet mit den Eingaben: `Hund`, `Katze`, `Maus`, `stopp`

Lösung

Ausgabe:

```
Wort (stopp zum Beenden): «Hund»
Wort (stopp zum Beenden): «Katze»
Wort (stopp zum Beenden): «Maus»
Wort (stopp zum Beenden): «stopp»
Du hast 3 Wörter eingegeben.
```

Lösung

Wird sofort `stopp` eingegeben, läuft der Rumpf nie, und das Programm gibt `Du hast 0 Wörter eingegeben.` aus. Das ist richtig so.

Aufgabe 6: Ablaufprotokoll: Stellen zählen ●●○

Das Programm zählt, wie viele Stellen eine Zahl hat. Fülle das Ablaufprotokoll aus (– bedeutet: noch kein Wert) und gib an, was das Programm ausgibt.

```
zahl = 4096
stellen = 0
while zahl > 0:
    zahl = zahl // 10
    stellen = stellen + 1
print("Stellen:", stellen)
```

```
Stellen: 4
```

Schritt	Zeile	zahl	stellen	Schritt	Zeile	zahl	stellen
1	1			9	3		
2	2			10	4		
3	3			11	5		
4	4			12	3		
5	5			13	4		
6	3			14	5		
7	4			15	3		
8	5			16	6		

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

Schritt	Zeile	zahl	stellen	Schritt	Zeile	zahl	stellen
1	1	`4096`	–	9	3	`40`	`2`
2	2	`4096`	`0`	10	4	`4`	`2`
3	3	`4096`	`0`	11	5	`4`	`3`
4	4	`409`	`0`	12	3	`4`	`3`
5	5	`409`	`1`	13	4	`0`	`3`
6	3	`409`	`1`	14	5	`0`	`4`
7	4	`40`	`1`	15	3	`0`	`4`
8	5	`40`	`2`	16	6	`0`	`4`

Lösung

Ausgabe:

```
Stellen: 4
```

Lösung

`zahl // 10` schneidet in jedem Durchlauf die letzte Ziffer ab: aus 4096 wird 409, dann 40, dann 4, dann 0. Für jede abgeschnittene Ziffer wird `stellen` um 1 erhöht. Bei `0` ist keine Ziffer mehr übrig, und die Schleife endet. Das ist dasselbe Vorgehen wie beim Auftrag *Quersumme*.

Aufgabe 7: Papier falten ●●○

Ein Blatt Papier ist etwa 0,1 mm dick. Faltet man es einmal, ist der Stapel doppelt so dick. Stell dir vor, man könnte das Blatt beliebig oft falten.

1. Schreibe ein Programm, das berechnet, wie oft man falten müsste, bis der Stapel mindestens 1 Meter (1000 mm) dick ist. Gib die Anzahl der Faltungen und die Dicke aus.
2. Begründe, warum hier eine while-Schleife passt.

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

```
dicke = 0.1
faltungen = 0
while dicke < 1000:
    dicke = dicke * 2
    faltungen = faltungen + 1
print("Nach", faltungen, "Faltungen ist der Stapel",
      round(dicke, 1), "mm dick.")
```

```
Nach 14 Faltungen ist der Stapel 1638.4 mm dick.
```

Lösung

Ausgabe:

```
Nach 14 Faltungen ist der Stapel 1638.4 mm dick.
```

Lösung

`faltungen` zählt die Durchläufe und startet bei 0. Im Rumpf wird die Dicke verdoppelt und die Faltung gezählt. Nach 13 Faltungen sind es erst 819,2 mm, deshalb ist noch eine 14. Faltung nötig.

2. Eine while-Schleife passt, weil die Anzahl der Faltungen gerade das ist, was gesucht wird. Sie steht vor der Schleife nicht fest. Die Schleife läuft, **solange** der Stapel dünner als 1000 mm ist. Sie terminiert, weil die Dicke in jedem Durchlauf wächst.

Aufgabe 8: Die 3n+1-Folge ●●●

Mit einer Startzahl wird immer wieder so gerechnet:

- Ist die Zahl gerade, wird sie halbiert.
- Ist sie ungerade, wird sie mit 3 multipliziert und dann 1 addiert.

Das wird wiederholt, bis die Zahl 1 erreicht ist. Aus 6 wird zum Beispiel 3, dann 10, 5, 16, 8, 4, 2 und 1. Das sind 8 Schritte.

1. Schreibe ein Programm, das eine Startzahl einliest, jede neue Zahl ausgibt und am Ende die Anzahl der Schritte. Teste mit 6 und mit 7.
2. Was passiert bei der Startzahl `0`? Erkläre.

Deine Antwort:

…

Lösung

✅ Lösung

Lösung

```
1 zahl = int(input("Startzahl: "))
2 schritte = 0
3 while zahl != 1:
4     if zahl % 2 == 0:
5         zahl = zahl // 2
6     else:
7         zahl = 3 * zahl + 1
8     print(zahl)
9     schritte = schritte + 1
10 print("Schritte:", schritte)
```

```
Startzahl: 5
16
8
4
2
1
Schritte: 5
```

Lösung

Getestet mit den Eingaben: `6`

Lösung

Ausgabe:

```
Startzahl: «6»
3
10
5
16
8
4
2
1
Schritte: 8
```

Lösung

- **Zeile 3:** Die Schleife läuft, solange die 1 noch nicht erreicht ist. Wie viele Schritte das sind, weiß man vorher nicht, deshalb while.
- **Zeilen 4–7:** Im Rumpf steht eine Verzweigung. `zahl % 2 == 0` prüft, ob die Zahl gerade ist. Zum Halbieren nimmt man `//`, damit die Zahl ganz bleibt.
- **Zeilen 8–9:** Die neue Zahl wird ausgegeben und der Schritt gezählt. Beide Zeilen gehören zur Schleife, aber nicht mehr zur Verzweigung.

Mit der Startzahl 7 braucht das Programm 16 Schritte, die größte Zahl unterwegs ist 52.

2. Bei `0` entsteht eine **Endlosschleife**. 0 ist gerade, und `0 // 2` ergibt wieder `0`. Die Zahl wird also nie 1, und das Programm gibt endlos `0` aus. Abhilfe: Vorher mit einer while-Schleife prüfen, dass die Startzahl mindestens 1 ist.