DS 5 – Weitere Übungen

Die for-Schleife: Anweisungen wiederholen

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: Wie oft läuft die Schleife? ●○○

Bestimme für jeden Schleifenkopf, wie oft die Schleife durchlaufen wird und welchen Wert `i` im ersten und im letzten Durchlauf hat. Überprüfe am Computer, indem du in den Rumpf `print(i)` schreibst.

Schleifenkopf	Anzahl Durchläufe	erster Wert	letzter Wert
`for i in range(6):`			
`for i in range(1, 6):`			
`for i in range(3, 10):`			
`for i in range(0, 20, 5):`			
`for i in range(1, 2):`			
`for i in range(4, 4):`			

Lösung

✅ **Lösung**

Lösung

Schleifenkopf	Anzahl Durchläufe	erster Wert	letzter Wert
`for i in range(6):`	6	0	5
`for i in range(1, 6):`	5	1	5
`for i in range(3, 10):`	7	3	9
`for i in range(0, 20, 5):`	4	0	15
`for i in range(1, 2):`	1	1	1
`for i in range(4, 4):`	0	–	–

Der Stoppwert gehört **nie** dazu. Bei `range(start, stop)` gibt es deshalb `stop - start` Durchläufe, zum Beispiel `10 - 3 = 7`. Bei `range(0, 20, 5)` sind es die Werte 0, 5, 10 und 15 – die 20 ist schon der Stoppwert. `range(4, 4)` erzeugt keine Zahl, der Rumpf wird nie ausgeführt.

Aufgabe 2: Runde für Runde ●○○

Lies das Programm, **ohne** es auszuführen. Schreibe auf, was es ausgibt. Überprüfe deine Vorhersage danach am Computer.

```python
for i in range(1, 4):
    print("Runde", i)
    print("---")
print("Spiel vorbei")
```

```
Runde 1
---
Runde 2
---
Runde 3
---
Spiel vorbei
```

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

Ausgabe:

```
Runde 1
---
Runde 2
---
Runde 3
---
Spiel vorbei
```

Lösung

Die Zeilen 2 und 3 sind eingerückt und gehören zum Schleifenrumpf. Sie werden in jedem der drei Durchläufe ausgeführt, also auch der Trennstrich dreimal. Zeile 4 ist nicht eingerückt und wird erst nach der Schleife einmal ausgeführt.

Aufgabe 3: Einmaleins nach Wahl ●○○

Das Programm soll eine Zahl einlesen und ihr Einmaleins von 1 bis 10 ausgeben, zum Beispiel für 9 die Zeilen `9 * 1 = 9` bis `9 * 10 = 90`. Ergänze die drei Lücken `___`.

Lückencode

```python
zahl = int(input("Welche Reihe? "))
for i in range(___, ___):
    print(zahl, "*", i, "=", ___)
```

Lösung

✅ **Lösung**

Lösung

```
1 zahl = int(input("Welche Reihe? "))
2 for i in range(1, 11):
3     print(zahl, "*", i, "=", zahl * i)
```

```
Welche Reihe? 7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

Lösung

Getestet mit den Eingaben: `9`

Lösung

Ausgabe:

```
Welche Reihe? «9»
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90
```

Lösung

Damit die 10 noch dabei ist, muss der Stoppwert `11` sein. In der dritten Lücke wird gerechnet: `zahl * i` ohne Anführungszeichen. Das `"*"` davor steht in Anführungszeichen, es wird nur als Zeichen ausgegeben.

Aufgabe 4: Quadratzahlen ●○○

Schreibe ein Programm, das mit einer for-Schleife die Quadratzahlen von 1 bis 10 ausgibt. Jede Zeile soll so aussehen wie im Beispiel. Verwende den Operator `**`.

Ausgabe:

```
1 hoch 2 ist 1
2 hoch 2 ist 4
3 hoch 2 ist 9
```

Lösung

✅ **Lösung**

Lösung

```
for i in range(1, 11):
    print(i, "hoch 2 ist", i ** 2)
```

```
1 hoch 2 ist 1
2 hoch 2 ist 4
3 hoch 2 ist 9
4 hoch 2 ist 16
5 hoch 2 ist 25
6 hoch 2 ist 36
7 hoch 2 ist 49
8 hoch 2 ist 64
9 hoch 2 ist 81
10 hoch 2 ist 100
```

Lösung

Ausgabe:

```
1 hoch 2 ist 1
2 hoch 2 ist 4
3 hoch 2 ist 9
4 hoch 2 ist 16
5 hoch 2 ist 25
6 hoch 2 ist 36
7 hoch 2 ist 49
8 hoch 2 ist 64
9 hoch 2 ist 81
10 hoch 2 ist 100
```

Lösung

Die Zählvariable `i` läuft von 1 bis 10, deshalb `range(1, 11)`. In jedem Durchlauf wird `i ** 2` neu berechnet. Statt `i ** 2` ist auch `i * i` richtig.

Aufgabe 5: Fehler finden: Ungerade Zahlen addieren ●●○

Das Programm soll die Summe der ungeraden Zahlen von 1 bis 19 berechnen (1 + 3 + 5 + … + 19) und **einmal** ausgeben. Richtig wäre `Summe: 100`. Das Programm läuft ohne Fehlermeldung, enthält aber **drei Fehler**.

```
for i in range(1, 19, 2):
    summe = 0
    summe = summe + i
    print("Summe:", summe)
```

```
Summe: 1
Summe: 3
Summe: 5
Summe: 7
Summe: 9
Summe: 11
Summe: 13
Summe: 15
Summe: 17
```

1. Führe das Programm aus und beschreibe, was falsch ist.
2. Finde die drei Fehler und schreibe das Programm richtig auf.

Deine Antwort:

...

Lösung

✅ **Lösung**

Lösung

Das fehlerhafte Programm gibt neun Zeilen aus: `Summe: 1`, `Summe: 3`, `Summe: 5` bis `Summe: 17`.

1. **Zeile 1:** Der Stoppwert `19` gehört nicht dazu. Die Schleife endet bei 17. Richtig ist `range(1, 20, 2)` (oder `21`).
2. **Zeile 2:** `summe = 0` steht in der Schleife. Dann wird die Summe in jedem Durchlauf wieder auf 0 gesetzt. Der Startwert gehört **vor** die Schleife.
3. **Zeile 4:** `print` ist eingerückt. Dann wird nach jedem Durchlauf ausgegeben statt einmal am Ende.

Lösung

```
summe = 0
for i in range(1, 20, 2):
    summe = summe + i
print("Summe:", summe)
```

```
Summe: 100
```

Lösung

Ausgabe:

```
Summe: 100
```

Lösung

Solche Fehler nennt man **logische Fehler**: Python meldet nichts, aber das Ergebnis stimmt nicht. Man findet sie nur, wenn man das erwartete Ergebnis kennt und vergleicht.

Aufgabe 6: Ablaufprotokoll: Abziehen ●●○

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

```
wert = 10
for i in range(2, 8, 2):
    wert = wert - i
print("wert =", wert)
```

```
wert = -2
```

Schritt	Zeile	Anweisung	`i`	`wert`
1	1	`wert = 10`		
2	2	`for i in range(2, 8, 2):`		
3	3	`wert = wert - i`		
4	2	`for i in range(2, 8, 2):`		
5	3	`wert = wert - i`		
6	2	`for i in range(2, 8, 2):`		
7	3	`wert = wert - i`		
8	2	`for i in range(2, 8, 2):`		
9	4	`print("wert =", wert)`		

Deine Antwort:

...

Lösung

✅ **Lösung**

Lösung

Schritt	Zeile	Anweisung	i	wert
1	1	`wert = 10`	-	10
2	2	`for i in range(2, 8, 2):`	2	10
3	3	`wert = wert - i`	2	8
4	2	`for i in range(2, 8, 2):`	4	8
5	3	`wert = wert - i`	4	4
6	2	`for i in range(2, 8, 2):`	6	4
7	3	`wert = wert - i`	6	-2
8	2	`for i in range(2, 8, 2):`	6	-2
9	4	`print("wert =", wert)`	6	-2

Lösung

Ausgabe:

```
wert = -2
```

Lösung

`range(2, 8, 2)` liefert 2, 4 und 6. Die 8 ist der Stoppwert und gehört nicht dazu. Abgezogen werden also 2 + 4 + 6 = 12, und 10 − 12 = −2. Im Schritt 8 liefert `range` keine Zahl mehr, die Schleife endet.

Aufgabe 7: Sparschwein ●●○

Im Sparschwein liegt schon ein Startbetrag. Jede Woche kommt derselbe Betrag dazu. Schreibe ein Programm, das den Startbetrag, den Betrag pro Woche und die Anzahl der Wochen (alles ganze Zahlen) einliest. Es soll für jede Woche den Stand ausgeben.

So könnte es aussehen:

Ausgabe:

```
Startbetrag in Euro: «10»
Betrag pro Woche in Euro: «3»
Wie viele Wochen? «4»
Nach Woche 1 sind 13 Euro im Sparschwein.
Nach Woche 2 sind 16 Euro im Sparschwein.
Nach Woche 3 sind 19 Euro im Sparschwein.
Nach Woche 4 sind 22 Euro im Sparschwein.
```

Lösung

✅ **Lösung**

Lösung

```
stand = int(input("Startbetrag in Euro: "))
pro_woche = int(input("Betrag pro Woche in Euro: "))
wochen = int(input("Wie viele Wochen? "))
for woche in range(1, wochen + 1):
    stand = stand + pro_woche
    print("Nach Woche", woche, "sind", stand, "Euro im Sparschwein.")
```

```
Startbetrag in Euro: 12
Betrag pro Woche in Euro: 12
Wie viele Wochen? 4
Nach Woche 1 sind 24 Euro im Sparschwein.
Nach Woche 2 sind 36 Euro im Sparschwein.
Nach Woche 3 sind 48 Euro im Sparschwein.
Nach Woche 4 sind 60 Euro im Sparschwein.
```

Lösung

Getestet mit den Eingaben: `10`, `3`, `4`

Lösung

Ausgabe:

```
Startbetrag in Euro: «10»
Betrag pro Woche in Euro: «3»
Wie viele Wochen? «4»
Nach Woche 1 sind 13 Euro im Sparschwein.
Nach Woche 2 sind 16 Euro im Sparschwein.
Nach Woche 3 sind 19 Euro im Sparschwein.
Nach Woche 4 sind 22 Euro im Sparschwein.
```

Lösung

`stand` funktioniert wie eine Summenvariable. Sie startet aber nicht bei 0, sondern beim Startbetrag. Die Zählvariable heißt hier `woche`, damit man sieht, wofür sie steht. Damit die letzte Woche dabei ist, ist der Stoppwert `wochen + 1`. Weil `print` eingerückt ist, wird der Stand nach **jeder** Woche ausgegeben.

Aufgabe 8: Notendurchschnitt ●●●

Schreibe ein Programm, das zuerst einliest, wie viele Noten es gibt. Danach soll es mit einer for-Schleife jede Note einzeln einlesen. Am Ende soll es die Summe der Noten und den Durchschnitt ausgeben, auf zwei Nachkommastellen gerundet.

Teste mit den Noten 2, 3, 1 und 2 (Durchschnitt 2.0) und mit 2, 2 und 1 (Durchschnitt 1.67). Was passiert, wenn du am Anfang `0` eingibst? Verbessere dein Programm so, dass es dann eine Meldung ausgibt, statt abzubrechen.

> **Tipp:** Die Eingabe der einzelnen Noten gehört **in** den Schleifenrumpf. Dann wird sie in jedem Durchlauf neu ausgeführt. Den Durchschnitt berechnest du **nach** der Schleife: Summe geteilt durch Anzahl.

Lösung

✅ Lösung

Lösung

```python
anzahl = int(input("Wie viele Noten? "))
summe = 0
for i in range(anzahl):
    note = float(input("Nächste Note: "))
    summe = summe + note
print("Summe:", summe)
if anzahl > 0:
    print("Durchschnitt:", round(summe / anzahl, 2))
else:
    print("Ohne Noten gibt es keinen Durchschnitt.")
```

```
Wie viele Noten? 3
Nächste Note: 2
Nächste Note: 3
Nächste Note: 1
Summe: 6.0
Durchschnitt: 2.0
```

Lösung

Getestet mit den Eingaben: `3`, `2`, `2`, `1`

Lösung

Ausgabe:

```
Wie viele Noten? «3»
Nächste Note: «2»
Nächste Note: «2»
Nächste Note: «1»
Summe: 5.0
Durchschnitt: 1.67
```

Lösung

- **Zeile 3:** `range(anzahl)` sorgt für genau `anzahl` Durchläufe. Die Zählvariable `i` wird im Rumpf nicht gebraucht.
- **Zeilen 4–5:** In jedem Durchlauf wird eine Note eingelesen und zur Summe addiert.
- **Zeilen 7–10:** Bei `anzahl = 0` läuft die Schleife gar nicht, und `summe / anzahl` würde durch 0 teilen. Ohne die Verzweigung bricht das Programm mit einem `ZeroDivisionError` ab.

Eine for-Schleife passt hier, weil die Anzahl der Noten schon **vor** der Schleife feststeht.