DS 5 – Die for-Schleife: Anweisungen wiederholen

Informatik 10 – MINT-Zweig | Textuelle Programmierung und Datenstrukturen

Lernziele

Ich kann

- erklären, was eine Schleife ist und wann man eine for-Schleife verwendet,
- eine for-Schleife mit Schleifenkopf und eingerücktem Schleifenrumpf schreiben,
- mit `range` Zahlenfolgen erzeugen, auch mit eigenem Start und eigener Schrittweite,
- mit einer Schleife eine Summe berechnen und dabei `+=` verwenden,
- den Ablauf einer Schleife in einem Ablaufprotokoll Schritt für Schritt verfolgen.

🗣️ Sprachhilfe

- *„Das wird fünfmal gemacht."* → „Die Anweisungen im Schleifenrumpf werden fünfmal wiederholt."
- *„i wird immer eins mehr."* → „Der Wert der Zählvariablen `i` wird bei jedem Durchlauf um 1 erhöht."
- *„Da kommt 15 raus."* → „Die Schleife addiert die Zahlen von 1 bis 5 zur Summenvariablen. Nach der Schleife gibt das Programm den Wert 15 aus."

Worum es geht

Dein Programm soll die Zahlen von 1 bis 100 ausgeben. Bisher bräuchtest du 100 `print`-Zeilen. Besser wäre: „Wiederhole das 100-mal." Aus der Blockprogrammierung kennst du „wiederhole 10-mal". In Python heißt das **for-Schleife**.

1 Wiederholen statt abschreiben

```
for i in range(3):
    print("Hallo!")
```

```
Hallo!
Hallo!
Hallo!
```

- **Zeile 1:** der **Schleifenkopf**: „Mache das Folgende dreimal." Er endet mit einem **Doppelpunkt**.
- **Zeile 2: eingerückt**. Alles, was eingerückt unter dem Kopf steht, wird wiederholt. Das ist der **Schleifenrumpf**.

🧪 **Probiere es aus:** Schreibe statt `3` die Zahl `100`.

📌 **Definition:** Eine **Schleife** ist eine Kontrollstruktur, die einen Block von Anweisungen (den **Schleifenrumpf**) mehrfach ausführt. Jede Ausführung heißt **Durchlauf**. Bei einer **for-Schleife** (Zählschleife) steht die Anzahl der Durchläufe schon vor dem Start fest.

📌 **Merksatz:** Die **for-Schleife** verwendet man, wenn die **Anzahl der Wiederholungen vorher bekannt** ist.

2 Aufbau einer for-Schleife

Die Zählvariable

Im Kopf steht eine Variable, hier `i`, die **Zählvariable**. Bei jedem Durchlauf bekommt sie automatisch den nächsten Wert.

```
1 for i in range(1, 4):
2     print("Runde", i)
3 print("Fertig")
```

```
Runde 1
Runde 2
Runde 3
Fertig
```

- **Zeile 1:** `range(1, 4)` liefert der Reihe nach `1`, `2` und `3`.
- **Zeile 2:** Der Rumpf gibt in jedem Durchlauf den aktuellen Wert von `i` aus.
- **Zeile 3: Nicht** eingerückt. Sie wird erst ausgeführt, wenn die Schleife fertig ist, und nur einmal.

Stell dir eine Läuferin auf der Bahn vor: Sie läuft ihre Runden und zählt mit. Nach der letzten Runde geht sie zur Bank (Zeile 3).

Schritt	Zeile	Anweisung	i
1	1	`for i in range(1, 4):`	1
2	2	`print("Runde", i)`	1
3	1	`for i in range(1, 4):`	2
4	2	`print("Runde", i)`	2
5	1	`for i in range(1, 4):`	3
6	2	`print("Runde", i)`	3
7	1	`for i in range(1, 4):`	3
8	3	`print("Fertig")`	3

- Nach dem Rumpf springt Python **zurück zum Kopf**. Dort bekommt `i` den nächsten Wert.
- In Schritt 7 liefert `range` keine Zahl mehr. Die Schleife endet, `i` behält den Wert `3`.
- Der Kopf wird viermal ausgeführt, der Rumpf dreimal.

> 📌 **Merksatz:** Die **Zählvariable** bekommt bei jedem Durchlauf automatisch den nächsten Wert. Du musst sie nicht selbst erhöhen.

Einrückung: Was gehört zur Schleife?

🔮 **Vorhersage:** Die beiden Programme unterscheiden sich nur in der letzten Zeile. Was gibt jedes aus?

```
1 for i in range(3):
2     print("Ich übe Python.")
3     print("Das macht Spaß!")
```

```
Ich übe Python.
Das macht Spaß!
Ich übe Python.
Das macht Spaß!
Ich übe Python.
Das macht Spaß!
```

```python
for i in range(3):
    print("Ich übe Python.")
print("Das macht Spaß!")
```

```
Ich übe Python.
Ich übe Python.
Ich übe Python.
Das macht Spaß!
```

⚠️ Stolpersteine: Einrückung und Doppelpunkt

Führe beide Zellen aus und lies die letzte Zeile der Fehlermeldung.

```python
for i in range(3):
print(i)
```

```
  File "<Zelle 5>", line 2
    print(i)
    ^^^^^
IndentationError: expected an indented block after 'for' statement on line 1
```

```python
for i in range(3)
    print(i)
```

```
  File "<Zelle 6>", line 1
    for i in range(3)
                     ^
SyntaxError: expected ':'
```

1. `IndentationError: expected an indented block after 'for' statement on line 1`: Der Rumpf muss eingerückt sein.
2. `SyntaxError: expected ':'`: Am Ende des Kopfes fehlt der Doppelpunkt.

3 Zahlenfolgen mit `range`

`range` („Bereich“) erzeugt eine Folge ganzer Zahlen.

- **`range(stop)`** beginnt bei `0` und endet **vor** `stop`.
- **`range(start, stop)`** beginnt bei `start` und endet **vor** `stop`.
- **`range(start, stop, schritt)`** zählt mit der **Schrittweite** `schritt`, negativ heißt rückwärts.

Aufruf	erzeugt die Zahlen	Anzahl
`range(5)`	0, 1, 2, 3, 4	5
`range(1, 6)`	1, 2, 3, 4, 5	5
`range(3, 8)`	3, 4, 5, 6, 7	5
`range(0, 10, 2)`	0, 2, 4, 6, 8	5
`range(1, 10, 3)`	1, 4, 7	3
`range(10, 0, -1)`	10, 9, 8, 7, 6, 5, 4, 3, 2, 1	10
`range(5, 1)`	keine Zahl (die Schleife läuft gar nicht)	0

> 📌 **Merksatz:** `range(n)` erzeugt `0, 1, …, n-1`. `range(start, stop)` erzeugt `start` bis `stop-1`. `range(start, stop, schritt)` erlaubt eine eigene Schrittweite.

> 💡 **Tipp:** `stop` ist wie die Ziellinie: Man zählt bis kurz davor. Bei `range(start, stop)` gibt es genau `stop - start` Zahlen.

Zahlen ausgeben und das Einmaleins

🔮 **Vorhersage:** Überlege zuerst, was das Programm ausgibt.

```python
for i in range(1, 6):
    print(i)
print("---")
for i in range(1, 11):
    print("7 *", i, "=", 7 * i)
```

```
1
2
3
4
5
---
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
```

- **Zeilen 1–2:** Für „1 bis 5“ braucht man `range(1, 6)`.
- **Zeile 3:** nicht eingerückt, wird einmal ausgegeben.
- **Zeilen 4–5:** `7 * i` wird in jedem Durchlauf neu ausgerechnet. Beide Schleifen dürfen `i` benutzen.

Rückwärts zählen

```python
for i in range(5, 0, -1):
    print(i)
print("Start!")
```

```
5
4
3
2
1
Start!
```

> ⚠️ **Stolperstein:** `range(1, 10)` erzeugt die Zahlen von 1 bis **9**. Python meldet keinen Fehler. Prüfe immer die erste und die letzte Zahl der Ausgabe. Für „1 bis 10“ schreibst du `range(1, 11)`.

🔮 Übung 1 – range-Detektiv

1. Welche Zahlen erzeugen die Aufrufe? Erst im Kopf, dann in der Codezelle prüfen.
2. Gib zu den Zahlenfolgen einen passenden `range`-Aufruf an.

Aufruf	erzeugt die Zahlen
`range(4)`	
`range(2, 7)`	
`range(5, 30, 5)`	
`range(9, 3, -2)`	

Zahlenfolge	Aufruf
1, 2, 3, 4, 5, 6, 7, 8	
10, 20, 30, 40, 50	
3, 2, 1, 0	

```python
# Prüfe hier jeden Aufruf, indem du die Zahl in range(...) änderst
for i in range(4):
    print(i)
```

```
0
1
2
3
```

✏️ Übung 2 – Zahlenreihe

Gib alle geraden Zahlen von 2 bis 20 aus. Tipp: Schrittweite nutzen. Prüfe die erste und die letzte Zahl!

Lückencode

```python
for i in range(___, ___, ___):
    print(i)
```

4 Summen bilden

Denk an eine Kasse: Am Anfang steht die Anzeige auf 0. Bei jedem Artikel kommt sein Preis dazu. Am Ende steht der Gesamtbetrag da. Im Programm brauchst du dafür eine **Summenvariable** (Fachwort **Akkumulator**, lateinisch *accumulare* = anhäufen).

```python
summe = 0
for i in range(1, 6):
    summe = summe + i
print("Summe:", summe)
```

```
Summe: 15
```

- **Zeile 1:** Startwert `0`, die Kasse ist leer.
- **Zeile 3:** Der aktuelle Wert von `i` kommt zur bisherigen Summe dazu, wie `alter = alter + 1`.
- **Zeile 4:** nicht eingerückt, die Summe wird einmal am Ende ausgegeben.

Lies im Ablaufprotokoll die Spalte `summe` von oben nach unten: 0, 1, 3, 6, 10, 15.

Schritt	Zeile	Anweisung	i	summe
1	1	`summe = 0`	-	0
2	2	`for i in range(1, 6):`	1	0
3	3	`summe = summe + i`	1	1
4	2	`for i in range(1, 6):`	2	1
5	3	`summe = summe + i`	2	3
6	2	`for i in range(1, 6):`	3	3
7	3	`summe = summe + i`	3	6
8	2	`for i in range(1, 6):`	4	6
9	3	`summe = summe + i`	4	10
10	2	`for i in range(1, 6):`	5	10
11	3	`summe = summe + i`	5	15
12	2	`for i in range(1, 6):`	5	15
13	4	`print("Summe:", summe)`	5	15

> 📌 **Merksatz:** Eine Summe berechnet man in drei Schritten: **vor** der Schleife `summe = 0`, **in** der Schleife `summe = summe + zahl`, **nach** der Schleife das Ergebnis ausgeben.

Die Kurzschreibweise `+=`

`summe += i` heißt „`summe` wird um `i` erhöht“ und bewirkt dasselbe wie `summe = summe + i`.

Kurzschreibweise	bedeutet	Beispiel (vorher `x = 10`)
`x += 3`	`x = x + 3`	`x` ist danach `13`
`x -= 3`	`x = x - 3`	`x` ist danach `7`
`x *= 3`	`x = x * 3`	`x` ist danach `30`

⚠️ Stolpersteine beim Summieren

🔮 **Vorhersage:** Was geben die beiden Programme aus? Vergleiche mit dem richtigen Programm oben.

```python
for i in range(1, 6):
    summe = 0
    summe = summe + i
print("Summe:", summe)
```

```
Summe: 5
```

```python
summe = 0
for i in range(1, 6):
    summe = summe + i
    print("Summe:", summe)
```

```
Summe: 1
Summe: 3
Summe: 6
Summe: 10
Summe: 15
```

1. Der Startwert steht **in** der Schleife: Die Summe wird jedes Mal auf 0 gesetzt, am Ende steht nur die letzte Zahl 5 darin.

2. Die Ausgabe ist eingerückt: Nach jedem Durchlauf kommt eine Zwischensumme. Das ist zum Nachprüfen nützlich, ist aber nur das Ergebnis gefragt, gehört `print` nach der Schleife.

Die Obergrenze eingeben lassen

`range` arbeitet nur mit ganzen Zahlen, deshalb `int(...)`. Damit `n` selbst mitgezählt wird, ist der Stoppwert `n + 1`.

```
n = int(input("Bis zu welcher Zahl soll ich addieren? "))
summe = 0
for i in range(1, n + 1):
    summe += i
print("Die Summe von 1 bis", n, "ist", summe)
```

```
Bis zu welcher Zahl soll ich addieren? 10
Die Summe von 1 bis 10 ist 55
```

⚠️ **Stolperstein:** Ohne `int(...)` ist `n` ein Text: `TypeError: 'str' object cannot be interpreted as an integer`. Mit `float` heißt es `'float'`. `range` braucht **ganze Zahlen**.

🔮 Übung 3 – Ablaufprotokoll

Fülle das Ablaufprotokoll aus, **ohne** das Programm auszuführen (– heißt: noch kein Wert). Gib dann an, was es ausgibt. Prüfe danach.

```
x = 0
for i in range(3):
    x = x + 2 * i
print("x =", x)
```

```
x = 6
```

Schritt	Zeile	Anweisung	i	x
1	1	`x = 0`	-	
2	2	`for i in range(3):`		
3	3	`x = x + 2 * i`		
4	2	`for i in range(3):`		
5	3	`x = x + 2 * i`		
6	2	`for i in range(3):`		
7	3	`x = x + 2 * i`		
8	2	`for i in range(3):`		
9	4	`print("x =", x)`		

Ausgabe: …

📝 Übungen: Schritt für Schritt selbstständig

Das **Musterbeispiel** ist ganz vorgemacht. Bei Übung 4 füllst du Lücken `___`, bei Übung 5 steht nur noch ein Gerüst aus Kommentaren, die Zusatzaufgabe schreibst du allein.

Schritt	Frage
1	**Vor der Schleife:** Welche Variable braucht einen Startwert?
2	**Schleifenkopf:** Welche Zahlen soll `i` annehmen? Erste und letzte Zahl prüfen!
3	**Rumpf:** Was passiert in jedem Durchlauf?
4	**Nach der Schleife:** Was wird einmal am Ende ausgegeben?

🧩 Musterbeispiel (vorgemacht) – Summe der geraden Zahlen

Aufgabe: Berechne die Summe der geraden Zahlen von 2 bis 10, also 2 + 4 + 6 + 8 + 10.

1. **Vor der Schleife:** `summe = 0`.
2. **Schleifenkopf:** `i` soll 2, 4, 6, 8, 10 sein. Start 2, Schrittweite 2, und der Stoppwert muss **größer** als 10 sein: `range(2, 11, 2)`.
3. **Rumpf:** `summe += i`.
4. **Nach der Schleife:** nicht eingerückt `print`.

```python
# Schritt 1: vor der Schleife
summe = 0

# Schritt 2 und 3: Schleifenkopf und Rumpf
for i in range(2, 11, 2):
    summe += i

# Schritt 4: nach der Schleife
print("Summe der geraden Zahlen von 2 bis 10:", summe)
```

```
Summe der geraden Zahlen von 2 bis 10: 30
```

Kontrolle von Hand: 2 + 4 + 6 + 8 + 10 = 30.

📝 Übung 4 – Summe von 1 bis 100 (Lücken füllen)

Berechne die Summe aller Zahlen von 1 bis 100, ohne eine fertige Summenfunktion. Ersetze jedes `___`. Kontrolle: Wer `4950` erhält, hat die 100 vergessen.

Lückencode

```python
summe = ___
for i in range(___, ___):
    summe ___ i
print("Summe:", ___)
```

📝 Übung 5 – Quadratzahlen (mit Gerüst)

Lies eine Zahl `n` ein. Gib für jede Zahl von 1 bis `n` ihre Quadratzahl aus und am Ende die Summe aller Quadratzahlen. Ersetze die Kommentare durch Programmzeilen.

```
Bis zu welcher Zahl? 4
1 zum Quadrat ist 1
2 zum Quadrat ist 4
3 zum Quadrat ist 9
4 zum Quadrat ist 16
Summe der Quadratzahlen: 30
```

```
# n als Ganzzahl einlesen
# Summenvariable mit Startwert
# Schleife: i von 1 bis n (Stoppwert beachten!)
#     Quadratzahl ausgeben
#     Quadratzahl zur Summe addieren
# nach der Schleife: Summe ausgeben
```

🚀 Zusatzaufgabe – Sternendreieck (selbstständig)

> 🧰 **Werkzeug:** Ein Text mal eine ganze Zahl hängt den Text so oft hintereinander: `"ab" * 3` ergibt `ababab`, `"-" * 20` ergibt 20 Striche. Die Zahl darf auch die Zählvariable sein.

Gib mit einer for-Schleife dieses Muster aus:

```
*
**
***
****
*****
```

Für Schnelle: Gib das Dreieck auch auf dem Kopf aus.

📌 Das Wichtigste in Kürze

- Die **for-Schleife** wiederholt den eingerückten **Rumpf**, wenn die Anzahl der Durchläufe vorher feststeht.
- Die **Zählvariable** bekommt bei jedem Durchlauf automatisch den nächsten Wert.
- `range(stop)`, `range(start, stop)`, `range(start, stop, schritt)`: `stop` gehört **nie** dazu.
- Summe: vor der Schleife `summe = 0`, in der Schleife `summe += zahl`, danach ausgeben.
- `range` braucht ganze Zahlen, Eingaben also mit `int(...)` umwandeln.

Zum Weiterüben: `DS05_Weitere_Uebungen.ipynb`.

Lösung

✅ Lösungen

Lösung Übung 1 – range-Detektiv:

Aufruf	erzeugt die Zahlen
`range(4)`	0, 1, 2, 3
`range(2, 7)`	2, 3, 4, 5, 6
`range(5, 30, 5)`	5, 10, 15, 20, 25
`range(9, 3, -2)`	9, 7, 5

Zahlenfolge	Aufruf
1, 2, 3, 4, 5, 6, 7, 8	`range(1, 9)`
10, 20, 30, 40, 50	`range(10, 51, 10)`
3, 2, 1, 0	`range(3, -1, -1)`

Bei der zweiten Tabelle gibt es mehrere richtige Antworten, z. B. `range(10, 60, 10)`. Bei der letzten Folge muss der Stoppwert `-1` sein, damit die `0` dabei ist.

Lösung

```python
# Kontrolle Übung 1
print(list(range(4)), list(range(2, 7)), list(range(5, 30, 5)), list(range(9, 3, -2)))
print(list(range(1, 9)), list(range(10, 51, 10)), list(range(3, -1, -1)))
```

```
[0, 1, 2, 3] [2, 3, 4, 5, 6] [5, 10, 15, 20, 25] [9, 7, 5]
[1, 2, 3, 4, 5, 6, 7, 8] [10, 20, 30, 40, 50] [3, 2, 1, 0]
```

Lösung

```python
# Lösung Übung 2 – Zahlenreihe
# Mit range(2, 20, 2) fehlt die 20. Auch richtig: for i in range(1, 11): print(2 * i)
for i in range(2, 21, 2):
    print(i)
```

```
2
4
6
8
10
12
14
16
18
20
```

Lösung

Lösung Übung 3 – Ablaufprotokoll:

Schritt	Zeile	Anweisung	i	x
1	1	`x = 0`	-	0
2	2	`for i in range(3):`	0	0
3	3	`x = x + 2 * i`	0	0
4	2	`for i in range(3):`	1	0
5	3	`x = x + 2 * i`	1	2
6	2	`for i in range(3):`	2	2
7	3	`x = x + 2 * i`	2	6
8	2	`for i in range(3):`	2	6
9	4	`print("x =", x)`	2	6

Ausgabe `x = 6`. `range(3)` beginnt bei **0**: 0 + 2 + 4 = 6.

Lösung

```python
# Lösung Übung 4 – Summe von 1 bis 100
# Gauß: 50 Paare (1 + 100, 2 + 99, …) mit der Summe 101, also 50 · 101 = 5050.
summe = 0
for i in range(1, 101):
    summe += i
print("Summe:", summe)
```

```
Summe: 5050
```

Lösung

```python
# Lösung Übung 5 – Quadratzahlen
n = int(input("Bis zu welcher Zahl? "))
summe = 0
for i in range(1, n + 1):
    print(i, "zum Quadrat ist", i ** 2)
    summe += i ** 2
print("Summe der Quadratzahlen:", summe)
```

```
Bis zu welcher Zahl? 10
1 zum Quadrat ist 1
2 zum Quadrat ist 4
3 zum Quadrat ist 9
4 zum Quadrat ist 16
5 zum Quadrat ist 25
6 zum Quadrat ist 36
7 zum Quadrat ist 49
8 zum Quadrat ist 64
9 zum Quadrat ist 81
10 zum Quadrat ist 100
Summe der Quadratzahlen: 385
```

Lösung

```python
# Lösung Zusatzaufgabe – Sternendreieck
# Mit range(5) wäre die erste Zeile leer ("*" * 0) und die letzte hätte nur 4 Sterne.
for i in range(1, 6):
    print("*" * i)
print()
for i in range(5, 0, -1):
    print("*" * i)
```

```
*
**
***
****
*****

*****
****
***
**
*
```