Programmieren – 5. Schleifen

Ingenieurinformatik Teil 1, Wintersemester 2026/27

David Straub

Warm-up: Was gibt der Code aus?

Der Benutzer tippt `3` ein. Aufschreiben, dann ausführen:

```
zahl = input("Zahl: ")
print(zahl * 2)
print(f"{10 / 4:.1f}")
```

Heute lernen Sie

- Code **wiederholen**, ohne ihn zu kopieren: `while` und `for`
- Robuste Programme: **Eingabe wiederholen, bis sie gültig ist**
- Schleifen **vorzeitig abbrechen**: `break`
- Endlosschleifen erkennen – und stoppen

`while`: wiederholen, solange …

Ohne Schleife wäre es so

```
print("Messung 1")
print("Messung 2")
print("Messung 3")
# ... und bei 1000 Messungen?
```

Kopierter Code ist der Anfang allen Übels. Es geht besser:

Die `while`-Schleife

```
i = 1
while i <= 3:
    print(f"Messung {i}")
    i = i + 1
```

- Solange die Bedingung `True` ist, läuft der eingerückte Block – **immer wieder**

- Gleiche Ausgabe wie eben – aber der Code wächst nicht mit der Anzahl: für 1000 Messungen ändert sich nur die `3`
- Von Hand verfolgen (so prüft man Schleifen!):

Durchlauf	`i`	`i <= 3` ?	Ausgabe
1	1	True	Messung 1
2	2	True	Messung 2
3	3	True	Messung 3
–	4	False	*(Ende)*

Kurzform für das Hochzählen: `i += 1` bedeutet `i = i + 1` – diese Kurzform sehen Sie überall in echtem Code.

Die Endlosschleife

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

- `i` ändert sich nie – die Bedingung bleibt für immer `True`
- Kein Fehler, keine Meldung: das Programm läuft einfach **endlos**
- Kennen Sie schon: die Batterie-Schleife aus Woche 1 war genau das!
- Stoppen in Jupyter: das **Stopp-Symbol** (Kernel unterbrechen)

Mini-Aufgabe (3 min)

```
i = 0
while i < 5:
    print(i)
    i += 1
```

1. Führen Sie den Code aus
2. Ändern Sie ihn so, dass nur bis `3` gezählt wird (letzte Ausgabe: `3`)
3. Ändern Sie ihn so, dass er bei `1` startet

Robuste Programme: Eingabe wiederholen, bis sie gültig ist

```
zahl = int(input("Zahl (1-100): "))
while zahl < 1 or zahl > 100:
    zahl = int(input("Ungültig! Bitte nochmal: "))
print(f"Ihre Zahl: {zahl}")
```

- Die Schleife läuft, **solange die Eingabe ungültig ist**
- Das `or` kennen Sie aus Woche 3 – „ungültig“ ist ein `or` aus Verstößen
- Nutzer tippen Unsinn – gute Programme fangen das freundlich ab. Dieses Muster gehört zum Handwerk

Debug-Aufgabe (3 min)

Dieser Countdown soll `5 4 3 2 1 Start!` ausgeben – stattdessen läuft er endlos. Finden Sie den Fehler und beheben Sie ihn:

```
countdown = 5
while countdown > 0:
    print(countdown)
print("Start!")
```

`for`: wiederholen mit Zähler

Die `for`-Schleife mit `range()`

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

- `range(5)` erzeugt die Zahlen `0, 1, 2, 3, 4` – **die 5 ist nicht dabei!**
- Start bei 0, wie beim Zählen in Python üblich
- `range(...)` ist übrigens wieder ein Funktionsaufruf

`range` mit Start und Stopp

```
for i in range(1, 6):
    print(i)
```

- `range(1, 6)` ⟶ `1, 2, 3, 4, 5` – Start dabei, Stopp nicht
- Es gibt auch eine Schrittweite: `range(0, 10, 2)` zählt in Zweierschritten

`while` oder `for`?

- **`for`**: die Anzahl der Wiederholungen steht vorher fest („10-mal“, „für jede Zahl von 1 bis 100“)
- **`while`**: wiederholen, solange eine Bedingung gilt („bis die Eingabe gültig ist“)
- Jede `for`-Schleife lässt sich als `while`-Schleife schreiben – das üben wir in Woche 8

Abbrechen mit `break`

```
for i in range(10):
    if i == 4:
        break
    print(i)
```

- `break` beendet die Schleife **sofort** – der Rest der Durchläufe entfällt

Ausgabe ohne Zeilenumbruch

```
for i in range(5):
    print("#", end="")
```

- Normal macht `print` nach jeder Ausgabe einen Zeilenumbruch
- `end=""` unterdrückt ihn – die Ausgaben landen in **einer Zeile**: `#####`

Vorhersage-Aufgabe (2 min)

Aufschreiben, dann ausführen:

```
for i in range(1, 8):
    if i == 5:
        break
    print(i, end=" ")
```

Transfer

Aufgabe: Balkenanzeige (Denkphase: 5 min, auf Papier)

Schreiben Sie ein Programm:

1. Eine Zahl `n` zwischen 1 und 20 **einlesen** – ungültige Eingaben wiederholen (das neue Muster von eben!)
2. Dann eine Zeile mit genau `n` Rauten **ausgeben** (`print`), z. B. für `n = 6` :

```
######
```

Notieren Sie zuerst: Welche Schleife für die Eingabe? Welche für die Rauten? Warum?

Zusatz für Schnelle: Geben Sie hinter den Rauten die Zahl aus (`print`), z. B. `###### 6`

Gemeinsam live

Wir entwickeln die Lösung jetzt zusammen.

Mini-Check

Ohne Computer – was kommt heraus?

1. `range(4)` – welche Zahlen sind das?
2. Warum läuft `while i < 3: print(i)` endlos?
3. Was bewirkt `end=""` bei `print`?
4. Eingabe validieren, bis sie passt – `for` oder `while`? Warum?

Bis nächste Woche!

Nächste Woche: **Funktionen** – eigene Bausteine bauen

- Üben: KI-Quizze auf OneTutor: https://hm.onetutor.ai/
- Fragen: jederzeit im Matrix-Chat