DS 6 – Die while-Schleife: Wiederholen, solange eine Bedingung gilt

Informatik 10 – MINT-Zweig | Textuelle Programmierung und Datenstrukturen

Lernziele

Ich kann

- eine while-Schleife mit Bedingung und eingerücktem Schleifenrumpf schreiben,
- den Ablauf einer while-Schleife in einem Ablaufprotokoll darstellen,
- eine Endlosschleife erkennen, abbrechen und ihre Ursache beheben,
- Eingaben so lange wiederholen lassen, bis sie gültig sind,
- Zufallszahlen mit `random.randint` erzeugen,
- begründet entscheiden, ob eine for- oder eine while-Schleife besser passt.

🗣️ Sprachhilfe

- *„Das läuft, bis man 0 eingibt."* → „Die Anweisungen im Schleifenrumpf werden wiederholt, solange die eingegebene Zahl ungleich 0 ist."
- *„Dann hört es auf."* → „Sobald die Bedingung `zahl != 0` falsch ist, terminiert die Schleife, und das Programm gibt die Summe aus."
- *„Das hängt sich auf."* → „Die Schleife terminiert nicht, weil der Wert der Variablen `i` im Schleifenrumpf nicht erhöht wird."

Worum es geht

Mit der for-Schleife wiederholst du eine feste Anzahl von Malen. Oft weiß man das vorher nicht: Beim Würfelspiel würfelst du, **bis** eine Sechs kommt. Ein Geldautomat fragt die PIN, **solange** sie falsch ist. Dafür gibt es die **while-Schleife**.

1 Wiederholen, solange eine Bedingung wahr ist

while heißt „solange". Die Bedingung kennst du von `if`: ein Ausdruck, der `True` oder `False` ergibt.

```
1 i = 1
2 while i <= 3:
3     print(i)
4     i = i + 1
5 print("Ende")
```

```
1
2
3
Ende
```

- **Zeile 1:** Startwert. Das musst du bei while selbst erledigen.
- **Zeile 2:** `while`, Bedingung, Doppelpunkt. Vor **jedem** Durchlauf prüft Python, ob `i <= 3` wahr ist.
- **Zeilen 3–4:** der Rumpf. Die Erhöhung in Zeile 4 ist wichtig: Ohne sie bliebe `i` immer `1`.
- **Zeile 5:** wird ausgeführt, sobald die Bedingung falsch ist.

Schritt	Zeile	Anweisung	`i`
1	1	`i = 1`	`1`
2	2	`while i <= 3:`	`1`
3	3	`print(i)`	`1`
4	4	`i = i + 1`	`2`
5	2	`while i <= 3:`	`2`
6	3	`print(i)`	`2`
7	4	`i = i + 1`	`3`
8	2	`while i <= 3:`	`3`
9	3	`print(i)`	`3`
10	4	`i = i + 1`	`4`
11	2	`while i <= 3:`	`4`
12	5	`print("Ende")`	`4`

In Schritt 11 ist `4 <= 3` falsch, die Schleife **endet**. Am Ende hat `i` den Wert `4`, obwohl zuletzt `3` ausgegeben wurde.

> 📌 **Definition:** Eine **while-Schleife** (bedingte Schleife) wiederholt ihren Rumpf, **solange** eine Bedingung wahr ist. Die Bedingung wird **vor** jedem Durchlauf geprüft. Ist sie schon beim ersten Mal falsch, wird der Rumpf kein einziges Mal ausgeführt.

> 📌 **Merksatz:** Die **while-Schleife** verwendet man, wenn die **Anzahl der Wiederholungen vorher nicht bekannt** ist. Syntax: `while Bedingung:` und darunter der eingerückte Block.

2 Zahlen addieren, bis 0 eingegeben wird

Hier hilft die for-Schleife nicht: Wie viele Zahlen es sind, entscheidet die Nutzerin beim Eintippen.

```
1 summe = 0
2 zahl = int(input("Zahl eingeben (0 zum Beenden): "))
3 while zahl != 0:
4     summe = summe + zahl
5     zahl = int(input("Zahl eingeben (0 zum Beenden): "))
6 print("Summe aller Eingaben:", summe)
```

```
Zahl eingeben (0 zum Beenden): 4
Zahl eingeben (0 zum Beenden): 7
Zahl eingeben (0 zum Beenden): 0
Summe aller Eingaben: 11
```

- **Zeile 2:** Die **erste** Zahl wird **vor** der Schleife eingelesen. Sonst hätte `zahl` in Zeile 3 noch keinen Wert.
- **Zeile 3:** Solange die Zahl nicht 0 ist, wird der Rumpf ausgeführt.
- **Zeile 5:** Die **nächste** Zahl wird eingelesen. Diese Zeile sorgt dafür, dass sich die Bedingung ändern kann.

Ablaufprotokoll für die Eingaben 4, 7, 2, 0:

Schritt	Zeile	zahl	summe
1	1	-	0
2	2	4	0
3	3	4	0
4	4	4	4
5	5	7	4
6	3	7	4
7	4	7	11
8	5	2	11
9	3	2	11
10	4	2	13
11	5	0	13
12	3	0	13
13	6	0	13

> 💡 **Muster für Schleifen mit Eingaben:** 1. **vor** der Schleife die erste Eingabe einlesen, 2. im Kopf die Eingabe prüfen, 3. **am Ende** des Rumpfs die nächste Eingabe einlesen.

🧪 **Probiere es aus:** Gib gleich als Erstes `0` ein. Warum wird der Rumpf kein einziges Mal ausgeführt?

3 Endlosschleifen

Bei while bist **du** dafür verantwortlich, dass die Bedingung irgendwann falsch wird. Ein Countdown soll von 3 herunterzählen. 🔮 **Findest du den Fehler, bevor du die Zelle startest?**

Führe die Zelle aus und **brich sie ab**: in Jupyter mit der Stopp-Schaltfläche (Quadrat) oben, in VS Code mit „Interrupt“. Links neben der Zelle steht so lange ein Sternchen `[*]`.

```
1 zahl = 3
2 while zahl > 0:
3     print(zahl)
4     zahl = zahl + 1
5 print("Start!")
```

```
3
4
5
6
7
8
9
10
11
12
13
14
…
```

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

In Zeile 4 wird `zahl` erhöht statt verringert, `zahl > 0` bleibt für immer wahr. Richtig wäre `zahl = zahl - 1`. Beim Abbrechen meldet Python `KeyboardInterrupt`: Das ist kein Fehler im Code, nur die Bestätigung, dass du

angehalten hast.

> 📌 **Definition:** Eine **Endlosschleife** ist eine Schleife, deren Bedingung nie falsch wird. Sie **terminiert nicht**.

> 📌 **Merksatz:** Die Bedingung muss sich **innerhalb der Schleife verändern** können, sonst entsteht eine Endlosschleife.

So findest du die Ursache:

1. Welche Variablen stehen in der Bedingung?
2. Wird mindestens eine davon **im Rumpf** verändert (`i = i + 1`, neues `input(...)`)?
3. In die **richtige Richtung**?
4. Steht die Änderung wirklich **eingerückt** im Rumpf?

🐞 Übung 1 – Endlosschleife reparieren

Der Code enthält eine Endlosschleife. Ändere **nicht** die Bedingung `i < 10`, sondern behebe den Fehler im Rumpf. Kontrolle: Ausgabe 0 bis 9. (Vorsicht beim Starten der unveränderten Zelle: abbrechen!)

```
1 i = 0
2 while i < 10:
3     print(i)
```

```
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).

4 Eingaben prüfen: Wiederholen, bis die Eingabe stimmt

Das Programm akzeptiert nur Schulnoten von 1 bis 6. Wie oft sich jemand vertippt, weiß man vorher nicht.

```
1 note = int(input("Gib deine Note ein (1 bis 6): "))
2 while note < 1 or note > 6:
3     print("Ungültig! Erlaubt sind nur 1 bis 6.")
4     note = int(input("Gib deine Note ein (1 bis 6): "))
5 print("Gespeichert: Note", note)
```

```
Gib deine Note ein (1 bis 6): 2
Gespeichert: Note 2
```

- **Zeile 2:** Die Bedingung beschreibt eine **ungültige** Eingabe. Solange das zutrifft, wird wiederholt.
- **Zeile 5:** Hier kommt das Programm nur an, wenn die Note gültig ist.

> 💡 **Tipp:** Frage dich: „**Wann muss ich noch einmal fragen?**“ Die Antwort ist die Bedingung der while-Schleife.

5 Zufallszahlen

> 🧰 **Werkzeug:** Das **Modul** `random` ist eine Sammlung von Befehlen rund um den Zufall. Schreibe **einmal am Anfang** `import random`. Dann liefert `random.randint(a, b)` eine zufällige ganze Zahl von `a` bis `b`. Anders als bei `range` gehören **beide** Grenzen dazu: `random.randint(1, 6)` ist ein Würfel.

Das Programm würfelt, bis eine Sechs fällt, und zählt die Würfe. Starte die Zelle mehrmals.

```
1 import random
2 wurf = random.randint(1, 6)
3 anzahl = 1
4 print("Gewürfelt:", wurf)
5 while wurf != 6:
6     wurf = random.randint(1, 6)
7     anzahl += 1
8     print("Gewürfelt:", wurf)
9 print("Endlich eine Sechs! Anzahl der Würfe:", anzahl)
```

```
Gewürfelt: 1
Gewürfelt: 3
Gewürfelt: 5
Gewürfelt: 5
Gewürfelt: 6
Endlich eine Sechs! Anzahl der Würfe: 5
```

- **Zeilen 2–4:** Der erste Wurf wird vor der Schleife ausgeführt, gezählt und ausgegeben.
- **Zeilen 6–8:** Neu würfeln, Zähler erhöhen, ausgeben. Durch das neue Würfeln kann sich die Bedingung ändern.

6 for oder while?

🔮 **Vorhersage:** Geben beide Zellen dasselbe aus?

```
for i in range(1, 4):
    print(i)
```

```
1
2
3
```

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

```
1
2
3
```

Ja. Die for-Schleife ist aber kürzer und sicherer: Startwert, Bedingung und Erhöhung stecken alle in `range(1, 4)`. Für „würfle bis zur Sechs“ passt for nicht, weil `range` eine feste Anzahl braucht. Die entscheidende Frage: **Steht beim Start der Schleife fest, wie oft wiederholt wird?** (Das gilt auch, wenn die Anzahl vorher eingegeben wurde.)

Situation	Schleife	Begründung
Zahlen von 1 bis 100 ausgeben	`for`	genau 100 Durchläufe
Summe von 1 bis `n`, `n` wird eingegeben	`for`	beim Start der Schleife steht `n` fest
Zahlen addieren, bis 0 eingegeben wird	`while`	Ende hängt von der Eingabe ab
Eingabe wiederholen, bis sie gültig ist	`while`	Anzahl der Fehlversuche unbekannt
würfeln, bis eine Sechs fällt	`while`	Ende hängt vom Zufall ab

> 📌 **Merksatz:** Anzahl der Durchläufe vorher bekannt → `for`. Ende hängt von einer Bedingung oder Eingabe ab → `while`.

✏️ Übung 2 – for oder while?

Entscheide und begründe kurz. Du musst die Programme nicht schreiben.

Situation	for oder while?	Begründung
a) 20-mal „Hallo“ ausgeben		
b) Wörter einlesen, bis „stopp“ eingegeben wird		
c) das Einmaleins einer eingegebenen Zahl ausgeben		
d) 100 € wachsen jedes Jahr um 5 %: Nach wie vielen Jahren sind es mindestens 200 €?		

📝 Übungen: Schritt für Schritt selbstständig

Das **Musterbeispiel** ist ganz vorgemacht. Bei Übung 3 füllst du Lücken `___`, bei Übung 4 steht nur noch ein Gerüst aus Kommentaren, die Zusatzaufgabe schreibst du allein.

Schritt	Frage
1	**Vor der Schleife:** Welche Variablen brauchen einen Startwert? Welche erste Eingabe?
2	**Bedingung:** Wann muss die Schleife **noch einmal** laufen?
3	**Rumpf:** Was passiert pro Durchlauf? Was ändert die Bedingung (neue Eingabe, Erhöhung)?
4	**Nach der Schleife:** Was wird einmal am Ende ausgegeben?

🧩 Musterbeispiel (vorgemacht) – Wörter zählen

Aufgabe: Das Programm liest Wörter ein, bis „stopp“ eingegeben wird. Am Ende gibt es aus, wie viele Wörter es waren (ohne „stopp“).

1. **Vor der Schleife:** Zähler `anzahl = 0`, erstes Wort einlesen.
2. **Bedingung:** noch einmal, solange das Wort nicht „stopp“ ist: `wort != "stopp"`.
3. **Rumpf:** Zähler um 1 erhöhen, **am Ende** das nächste Wort einlesen.
4. **Nach der Schleife:** die Anzahl ausgeben.

```python
# Schritt 1: vor der Schleife
anzahl = 0
wort = input("Wort (stopp zum Beenden): ")

# Schritt 2: Bedingung
while wort != "stopp":
    # Schritt 3: Rumpf
    anzahl += 1
    wort = input("Wort (stopp zum Beenden): ")

# Schritt 4: nach der Schleife
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.
```

Testfälle: Hund, Katze, Maus, stopp → 3 Wörter. Sofort `stopp` → 0 Wörter, der Rumpf läuft kein einziges Mal.

📝 Übung 3 – Zahl erraten (Lücken füllen)

Die geheime Zahl ist im Code festgelegt. Man darf so lange raten, bis sie stimmt. Nach jedem Versuch kommt „zu hoch“ oder „zu niedrig“. Ersetze jedes `___`.

Kontrolle: 50 → zu hoch, 30 → zu niedrig, 42 → Richtig geraten!

Lückencode

```python
geheim = 42
versuch = int(input("Rate die Zahl: "))
while versuch ___ geheim:
    if versuch < geheim:
        print(___)
    else:
        print(___)
    versuch = ___
print("Richtig geraten!")
```

📝 Übung 4 – Quersumme (mit Gerüst)

Berechne mit einer while-Schleife die Quersumme einer eingegebenen Zahl. Beispiel: Die Quersumme von 384 ist 3 + 8 + 4 = 15.

> 💡 **Tipp:** `384 % 10` ergibt `4` (die letzte Ziffer), `384 // 10` ergibt `38` (die letzte Ziffer ist weg). Mit `38` bekommst du `8` und `3`, mit `3` bekommst du `3` und `0`. Bei `0` ist keine Ziffer mehr übrig: Das ist deine Abbruchbedingung.

Ersetze die Kommentare durch Programmzeilen. Weitere Kontrollwerte: 7 → 7, 1000 → 1, 99999 → 45.

```python
# Zahl einlesen
# Quersumme mit Startwert 0
# solange noch Ziffern übrig sind:
#     letzte Ziffer zur Quersumme addieren
#     letzte Ziffer abschneiden
# Quersumme ausgeben
```

🚀 Zusatzaufgabe – Ratespiel mit Zufallszahl (selbstständig)

Erweitere dein Programm aus Übung 3:

1. Die geheime Zahl wird mit `random.randint` zufällig zwischen 1 und 100 gewählt.

2. Das Programm zählt die Versuche und gibt am Ende aus, wie viele es waren.

> 💡 **Tipp zum Testen:** Lass die geheime Zahl vorübergehend mit `print(geheim)` ausgeben und lösche die Zeile danach.

📌 Das Wichtigste in Kürze

- `while Bedingung:` wiederholt den Rumpf, **solange** die Bedingung wahr ist. Sie wird **vor** jedem Durchlauf geprüft.
- Startwerte setzt du selbst, und im Rumpf muss sich die Bedingung ändern können, sonst **Endlosschleife**.
- Muster mit Eingaben: erste Eingabe vor der Schleife, nächste Eingabe am Ende des Rumpfs.
- `import random`, dann `random.randint(a, b)`: Zufallszahl von `a` bis `b`, beide Grenzen inklusive.
- Anzahl bekannt → `for`, Ende hängt von Bedingung ab → `while`.

Zum Weiterüben: `DS06_Weitere_Uebungen.ipynb`.