DS 7 – Übung: for- oder while-Schleife?

Informatik 10 – MINT-Zweig | Textuelle Programmierung und Datenstrukturen

Lernziele

Ich kann

- für eine Aufgabe begründet zwischen for- und while-Schleife wählen,
- ein Programm mit Schleife lesen und seine Ausgabe vorhersagen,
- Summen und Produkte mit einer Schleife berechnen,
- Eingaben mit einer while-Schleife wiederholen lassen,
- meine Programme mit sinnvoll gewählten Testfällen überprüfen.

🗣️ Sprachhilfe

- *„Ich nehme for, weil das besser ist."* → „Ich verwende eine for-Schleife, weil die Anzahl der Durchläufe beim Start feststeht: Es werden genau `n` Zahlen multipliziert."
- *„Ich nehme while, weil man nicht weiß, wie oft."* → „Ich verwende eine while-Schleife, weil das Ende von der Eingabe abhängt: Sie läuft, solange das Passwort falsch ist."
- *„Es hört auf."* → „Die Schleife terminiert, weil `zahl` in jedem Durchlauf größer wird."

Worum es geht

Du kennst zwei Schleifenarten. Heute übst du, beide sicher einzusetzen. Bei jeder Aufgabe überlegst du **zuerst**, welche Schleifenart besser passt, und begründest deine Wahl. Erst dann programmierst du.

1 Das brauchst du

	for-Schleife	while-Schleife
Wann?	Anzahl der Durchläufe steht beim Start fest	Ende hängt von einer Bedingung ab
Kopf	`for i in range(1, 6):`	`while zahl != 0:`
Startwert	steckt in `range`	vor der Schleife selbst setzen
Veränderung	automatisch	im Rumpf selbst programmieren
Gefahr	Stoppwert gehört nicht dazu	Endlosschleife

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

zahl = 1
while zahl < 10:
    print("zahl ist", zahl)
    zahl = zahl * 2
```

```
Durchlauf 1
Durchlauf 2
Durchlauf 3
zahl ist 1
zahl ist 2
zahl ist 4
zahl ist 8
```

📌 **Merksatz: for-Schleife:** Die Anzahl der Durchläufe steht vorher fest. **while-Schleife:** Das Ende hängt von einer Bedingung ab. Überlege bei jeder Aufgabe **zuerst**, welche Schleifenart passt!

Zwei Muster brauchst du immer wieder:

- **Zusammenzählen:** Summenvariable **vor** der Schleife auf `0`, **in** der Schleife erhöhen, **nach** der Schleife ausgeben.
- **Eingaben wiederholen:** erste Eingabe **vor** der while-Schleife, Bedingung prüft die Eingabe, nächste Eingabe **am Ende** des Rumpfs.

💡 **So testest du ein Programm mit Schleife:** nicht nur eine „normale“ Eingabe, sondern auch die kleinste erlaubte Zahl (oft 0 oder 1), eine Eingabe, bei der die Schleife **gar nicht** läuft, und eine, bei der sie sehr oft läuft. Rechne das erwartete Ergebnis **vorher** von Hand aus.

🔮 Übung 1 – Programm lesen

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

1. Was gibt es aus?
2. Wie oft wird der Schleifenrumpf ausgeführt? Lege dazu ein Ablaufprotokoll an.
3. Warum wird hier eine while-Schleife verwendet?

```
zahl = 1
while zahl < 50:
    zahl = zahl * 3
print(zahl)
```

```
81
```

Schritt	Zeile	Anweisung	zahl
1	1	`zahl = 1`	`1`
2	2	`while zahl < 50:`	
3	3	`zahl = zahl * 3`	
…			

Meine Antworten:

…

📝 Übungen: Schritt für Schritt selbstständig

Das **Musterbeispiel** ist ganz vorgemacht. Bei Übung 2 füllst du Lücken `___`, bei Übung 3 steht nur noch ein Gerüst aus Kommentaren, Übung 4 und die Zusatzaufgabe schreibst du allein.

Schritt	Frage
0	**Schleifenart wählen und begründen:** Steht beim Start fest, wie oft wiederholt wird?
1	**Vor der Schleife:** Welche Variablen brauchen einen Startwert? Welche erste Eingabe?
2	**Schleifenkopf:** `range(...)` oder Bedingung?
3	**Rumpf:** Was passiert pro Durchlauf?
4	**Nach der Schleife:** Was wird ausgegeben? Testfälle mit Grenzfällen.

🧩 Musterbeispiel (vorgemacht) – Notendurchschnitt

Aufgabe: Das Programm fragt, wie viele Noten es sind. Dann liest es die Noten ein und gibt den Durchschnitt aus.

0. **Begründung:** Eine for-Schleife passt. Sobald die Anzahl `n` eingegeben ist, steht fest, dass genau `n` Noten eingelesen werden.
1. **Vor der Schleife:** `n` einlesen, Summenvariable `summe = 0`.
2. **Schleifenkopf:** `for i in range(1, n + 1):`, damit `i` die Nummer der Note ist (1 bis `n`).
3. **Rumpf:** eine Note einlesen und zur Summe addieren.
4. **Nach der Schleife:** Durchschnitt `summe / n` ausgeben, auf zwei Stellen gerundet.

```python
# Schritt 1: vor der Schleife
n = int(input("Wie viele Noten? "))
summe = 0

# Schritt 2 und 3: Schleifenkopf und Rumpf
for i in range(1, n + 1):
    note = int(input("Note " + str(i) + ": "))
    summe += note

# Schritt 4: nach der Schleife
print("Durchschnitt:", round(summe / n, 2))
```

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

Testfälle: Noten 2, 3, 1 → Durchschnitt 2.0. Eine einzige Note 4 → 4.0 (die Schleife läuft genau einmal). Bei `n = 0` bricht das Programm mit `ZeroDivisionError` ab: Wie könnte man das mit einer Verzweigung abfangen?

📝 Übung 2 – Passwortabfrage (Lücken füllen)

Frage so lange nach einem Passwort, bis `informatik10` eingegeben wurde. Gib danach „Zugang gewährt“ aus.

1. **Begründe zuerst:** Welche Schleifenart passt?
2. Ersetze jedes `___`.

Testfälle: sofort richtig; `geheim`, dann `Informatik10`, dann `informatik10` (Python unterscheidet Groß- und Kleinschreibung).

Begründung: …

Lückencode

```python
passwort = ___("Passwort eingeben: ")
while passwort ___ "informatik10":
    passwort = ___("Falsch! Erneut versuchen: ")
print(___)
```

📝 Übung 3 – Fakultät (mit Gerüst)

In der Mathematik schreibt man *n*! („n Fakultät“) für das Produkt aller ganzen Zahlen von 1 bis *n*: 4! = 1 · 2 · 3 · 4 = 24. Für 0! legt man den Wert 1 fest. Die Fakultät gibt zum Beispiel an, auf wie viele Arten man *n* Personen in einer Reihe aufstellen kann.

1. **Begründe zuerst:** Welche Schleifenart passt?
2. Ersetze die Kommentare durch Programmzeilen.

> 💡 **Tipp:** Die Fakultät berechnest du fast wie eine Summe, nur wird multipliziert. Mit welchem Wert muss die Produktvariable **vor** der Schleife starten? Was passiert mit 0?

Kontrolle: 5 → `5! = 120`, 1 → `1! = 1`, 0 → `0! = 1`, 10 → `10! = 3628800`.

Begründung: …

```
# n als Ganzzahl einlesen
# Produktvariable mit passendem Startwert
# Schleife: i von 1 bis einschließlich n
#     Produkt mit i multiplizieren
# Ergebnis ausgeben, z. B. 5! = 120
```

📝 Übung 4 – Primzahl-Test (selbstständig)

Eine **Primzahl** ist eine natürliche Zahl größer als 1, die nur durch 1 und durch sich selbst teilbar ist: 2, 3, 5, 7, 11, 13, … Die 9 ist keine, weil sie durch 3 teilbar ist. Die 1 ist nach Festlegung keine Primzahl.

Prüfe, ob eine eingegebene Zahl eine Primzahl ist. Teste dazu, ob sie durch eine Zahl zwischen 2 und sich selbst minus 1 teilbar ist (`zahl % teiler == 0`).

> 💡 **Tipp:** Arbeite mit einer Variablen, die sich ein Ergebnis merkt. Nimm zuerst an, die Zahl **ist** eine Primzahl: `ist_primzahl = True`. Findest du einen Teiler, war die Annahme falsch. Denke an den Sonderfall 1.

Lege zuerst die Testfälle fest:

Eingabe	erwartete Ausgabe	Warum testen?
`7`		
`9`		
`2`		
`1`		
`97`		

🚀 Zusatzaufgabe – Sparziel (selbstständig)

Auf einem Sparkonto liegen 100 €. Jedes Jahr kommen 5 % Zinsen dazu, das Guthaben wird also mit 1.05 multipliziert. Nach wie vielen Jahren beträgt es zum ersten Mal mindestens 200 €? Gib die Anzahl der Jahre und das Guthaben auf Cent gerundet aus. Begründe deine Wahl der Schleifenart.

Für Schnelle: Beim Primzahl-Test prüft die for-Schleife weiter, auch wenn längst ein Teiler gefunden ist. Schreibe eine Fassung mit while, die beim ersten Teiler aufhört.

📌 Das Wichtigste in Kürze

- Erst die Schleifenart wählen und begründen: Steht die Anzahl beim Start fest → `for`, sonst → `while`.
- Summe startet bei `0`, Produkt startet bei `1`.
- Eingaben wiederholen: erste Eingabe vor der Schleife, nächste am Ende des Rumpfs.
- „Annahme-Variable“: `ist_primzahl = True` annehmen, beim Gegenbeispiel auf `False` setzen.
- Testen mit Grenzfällen: Schleife läuft gar nicht, genau einmal, sehr oft.

Zum Weiterüben: `DS07_Weitere_Uebungen.ipynb`.

Lösung

✅ Lösungen

Lösung Übung 1 – Programm lesen: Ausgabe `81`. Der Rumpf wird **viermal** ausgeführt.

Schritt	Zeile	Anweisung	`zahl`
1	1	`zahl = 1`	`1`
2	2	`while zahl < 50:`	`1`
3	3	`zahl = zahl * 3`	`3`
4	2	`while zahl < 50:`	`3`
5	3	`zahl = zahl * 3`	`9`
6	2	`while zahl < 50:`	`9`
7	3	`zahl = zahl * 3`	`27`
8	2	`while zahl < 50:`	`27`
9	3	`zahl = zahl * 3`	`81`
10	2	`while zahl < 50:`	`81`
11	4	`print(zahl)`	`81`

Die Ausgabe liegt über 50, weil die Bedingung erst **nach** dem Verdreifachen wieder geprüft wird. while passt, weil man vorher nicht weiß, wie oft man verdreifachen muss. Die Schleife terminiert, weil `zahl` in jedem Durchlauf größer wird.

Lösung Übung 2 – Begründung: while, denn niemand weiß vorher, wie viele Fehlversuche es gibt. Die Schleife läuft, solange das Passwort falsch ist. Ohne die neue Eingabe im Rumpf entstünde eine Endlosschleife.

Lösung

```
# Lösung Übung 2 – Passwortabfrage
passwort = input("Passwort eingeben: ")
while passwort != "informatik10":
    passwort = input("Falsch! Erneut versuchen: ")
print("Zugang gewährt")
```

```
Passwort eingeben: 1234
Falsch! Erneut versuchen: informatik10
Zugang gewährt
```

Lösung

Lösung Übung 3 – Begründung: for, denn sobald *n* eingegeben ist, steht fest, dass genau *n* Zahlen multipliziert werden. Die Zählvariable liefert gleich die Faktoren 1, 2, …, *n*. Die Produktvariable startet bei **1**: Mit 0 wäre jedes Produkt 0. Bei $n = 0$ läuft die Schleife gar nicht, der Startwert 1 bleibt stehen.

Lösung

```
# Lösung Übung 3 – Fakultät (kürzer: fakultaet *= i)
n = int(input("n eingeben: "))
fakultaet = 1
for i in range(1, n + 1):
    fakultaet = fakultaet * i
print(str(n) + "! =", fakultaet)
```

```
n eingeben: 5
5! = 120
```

Lösung

```python
# Kontrolle Übung 3 mit allen Testwerten
for n in [5, 1, 0, 10]:
    fakultaet = 1
    for i in range(1, n + 1):
        fakultaet *= i
    print(str(n) + "! =", fakultaet)
```

```
5! = 120
1! = 1
0! = 1
10! = 3628800
```

Lösung

```python
# Lösung Übung 4 – Primzahl-Test
# zahl > 1 ergibt True oder False: Für 1 und 0 startet ist_primzahl gleich mit False.
# range(2, 2) ist leer: Für 2 läuft die Schleife nicht, die Annahme True bleibt.
zahl = int(input("Zahl eingeben: "))
ist_primzahl = zahl > 1
for teiler in range(2, zahl):
    if zahl % teiler == 0:
        ist_primzahl = False
print("Primzahl:", ist_primzahl)
```

```
Zahl eingeben: 15
Primzahl: False
```

Lösung

```python
# Kontrolle Übung 4 mit allen Testwerten
for zahl in [7, 9, 2, 1, 97, 91]:
    ist_primzahl = zahl > 1
    for teiler in range(2, zahl):
        if zahl % teiler == 0:
            ist_primzahl = False
    print(zahl, "Primzahl:", ist_primzahl)
```

```
7 Primzahl: True
9 Primzahl: False
2 Primzahl: True
1 Primzahl: False
97 Primzahl: True
91 Primzahl: False
```

Lösung

```python
# Lösung Zusatzaufgabe – Sparziel
# while, denn die Anzahl der Jahre ist gerade das Gesuchte.
guthaben = 100
jahre = 0
while guthaben < 200:
    guthaben = guthaben * 1.05
    jahre += 1
print("Nach", jahre, "Jahren:", round(guthaben, 2), "Euro")
```

```
Nach 15 Jahren: 207.89 Euro
```

Lösung

```python
# Lösung für Schnelle – Primzahl-Test mit while: endet beim ersten Teiler (bei 91 schon bei 7)
zahl = int(input("Zahl eingeben: "))
teiler = 2
while teiler < zahl and zahl % teiler != 0:
    teiler += 1
if zahl > 1 and teiler == zahl:
    print(zahl, "ist eine Primzahl.")
else:
    print(zahl, "ist keine Primzahl.")
```

```
Zahl eingeben: 15
15 ist keine Primzahl.
```