

Kapitel 4: Algorithmische Grundstrukturen und ihre Darstellung

Bisher haben wir hauptsächlich Programme gesehen, bei denen der Programmcode Zeile für Zeile nacheinander ausgewertet wird. Mit Hilfe **algorithmischer Grundstrukturen** können wir diesem Verhalten abweichen.

Algorithmische Grundstrukturen steuern dabei den Ablauf eines Programms. Mit ihnen könnt ihr Anweisungsblöcke definieren, die nur unter einer bestimmten Bedingung ausgeführt werden oder auch solche, deren Ausführung mehrfach erfolgt. Hierfür unterscheiden wir **Sequenzen, Verzweigungen** und **Schleifen**, welche wir uns - inklusiv dreier **Darstellungsformen** - nachfolgend genauer anschauen.

4.8.6. Zählergesteuerte Schleife (kurz: Zählschleife)

Manchmal wissen wir **ganz genau**, wie oft etwas wiederholt werden soll. In einem solchen Fall können wir auf die direkte Überprüfung von Bedingungen verzichten und stattdessen eine Zählschleife verwenden.

Was brauchen wir dafür?

Wir benötigen einen **Start- und Endwert**: Den Beginn, von wo an wir anfangen zu zählen, und bis wohin. Weiterhin benötigen wir eine sogenannte Schrittweite. Entweder können wir im 1er-Schritt zählen (also z. B.: 0, 1, 2, ...) oder in beliebigen anderen Schrittweiten.

Die Struktur im **Programmcode** unterscheidet sich deutlich von den vorherigen Schleifen.

Am Beispiel:

```
In [ ]: for i in range(0, 5, 1):  
        print(i)
```

Erklärungen:

- Das Schlüsselwort für eine Zählschleife ist `for`.
- Nach dem Schlüsselwort folgt die Zählvariable, in diesem Fall `i`. Diese wird fortlaufend durch die Schleife verändert. Innerhalb der Schleife kann auf den aktuellen Wert zugegriffen werden. Die Zählvariable kann beliebig benannt werden, allerdings ist es üblich, die Buchstaben `i`, `j` oder `k` zu verwenden. Damit benötigen wir **keine** Variable mehr für den Zähler - wie bisher.
- Im Anschluss folgt das Schlüsselwort `in`. Das können wir "lesen" als: Die Zählvariable muss sich im nachfolgend definierten Wertebereich befinden.
- Mit Hilfe von `range(0, 4, 1)` definieren wir unseren Wertebereich. Die erste Zahl `0` definiert den Startwert. Die zweite Zahl `4` definiert unsere obere Grenze. Diese obere Grenze ist **exklusiv**. Die Zählvariable ist immer kleiner als dieser Wert. Die letzte Zahl `1` gibt die Schrittweite an. Wie im oberen Beispiel zu sehen, wird hier im 1er Schritt gezählt.

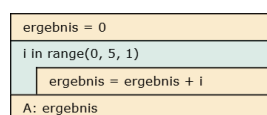
Im **Pseudocode** könnte man exemplarisch Folgendes formulieren:

```
WIEDERHOLE VON i = 0 BIS 5 IM 1er SCHRITT
    ergebnis = ergebnis + i
```

Die **Implementierung** ergibt sich wie folgt:

```
In [ ]: ergebnis = 0
        for i in range(0, 5, 1):
            ergebnis = ergebnis + i
        print(ergebnis)
```

Struktogramm:



Die geometrische Form entspricht der der kopfgesteuerten Schleife.

Als Bedingung schreiben wir direkt den Schleifenkopf `i in range(0, 5, 1)`. Das ist nicht unbedingt üblich, jedoch ermöglicht das Struktogramm-Tool die direkte Übersetzung des Struktogramms in ausführbaren Code.

Beispiel 1 - Wiederholte Satzausgabe:

```
In [ ]: satz = input('Welcher Satz soll wiederholt ausgegeben werden? Gib einen Satz ein: ')
        eingabe = input('Wie oft soll der Satz ausgegeben werden? Gib eine Anzahl ein: ')
        anzahl = int(eingabe)

        for i in range(anzahl):
            print(satz)

        print('Damit ist das Programm beendet.')

        # FÜHRE MICH AUS! :)
```

Wie an diesem Beispiel zu sehen ist, wird hier lediglich `range(anzahl)` verwendet. Dabei handelt es sich um eine **Kurzschreibweise**. In diesem Fall wird automatisch bei 0 begonnen zu zählen und die Schrittweite ist standardmäßig auf 1 festgelegt.

Beispiel 2 - Alle gerade Zahlen von 0 bis 10:

```
In [ ]: n = 0
        for i in range(0, 11, 2):
            print(i)

        # FÜHRE MICH AUS! :)
```

In diesem Beispiel wird deutlich, dass die **obere Grenze exklusiv** ist.

Wollen wir alle geraden Zahlen von 0 bis 10 ausgeben, **müssen** wir die obere Grenze auf 11 setzen. Weiterhin wurde die Schrittweite auf 2 festgesetzt, sodass nur jede zweite Zahl - und damit auch nur die geraden Zahlen - ausgegeben werden.



4.8.7. Übungen zur zählergesteuerten Schleife / Zähler Schleife

Aufgabe 1

Implementiere den folgenden Pseudocode.

```
Eingabe einer Zahl n
Setze die Variable Zähler auf 1
Setze die Variable Ergebnis auf 1
WIEDERHOLE VON i = 1 BIS n IM 1er SCHRITT
    ergebnis = ergebnis * zaehler
    zaehler = zaehler + 1
Ausgabe der Variable Ergebnis
```

```
In [2]: n = input('Gib eine Zahl n ein: ')
n = int(n)

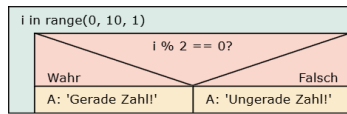
zaehler = 1
ergebnis = 1

for i in range(n):
    ergebnis = ergebnis * zaehler
    zaehler = zaehler + 1

print(ergebnis)
```

Aufgabe 2

Implementiere das folgende Struktogramm.



```
In [3]: for i in range(0, 10, 1):  
        if i % 2 == 0:  
            print('Gerade Zahl!')  
        else:  
            print('Ungerade Zahl!')
```

```
Gerade Zahl!  
Ungerade Zahl!  
Gerade Zahl!  
Ungerade Zahl!  
Gerade Zahl!  
Ungerade Zahl!  
Gerade Zahl!  
Ungerade Zahl!  
Gerade Zahl!  
Ungerade Zahl!
```

Aufgabe 3

Implementiere ein Programm, welches nach Eingabe einer oberen Grenze n , alle Zahlen von 0 bis n aufaddiert.

Bei der Eingabe der Zahl 3 ergibt sich beispielsweise das Ergebnis 6.

Gerechnet wird dabei: $0 + 1 + 2 + 3 = 6$.

Implementiere deine Lösung **zwingend** mit Hilfe einer **Zählschleife**!

```
In [5]: obere_grenze = input('Gib eine obere Grenze ein: ')
n = int(obere_grenze)

summe = 0

for i in range(0, n + 1):
    summe = summe + i

print(f'Die Quersumme von 0 bis {n} ergibt sich zu: {summe}')
```

Die Quersumme von 0 bis 3 ergibt sich zu: 6

Aufgabe 4

Implementiere ein Programm, welches die Zahlwörter eins, zwei, ... , sechs so oft ausgibt, wie es das Wort aussagt, also eins einmal, zwei zweimal.

Implementiere deine Lösung mit Hilfe einer **Zählschleife**! Ein **korrekt funktionierendes Programm** erzeugt die folgende Ausgabe:

```
eins
zwei zwei
drei drei drei
vier vier vier vier
fünf fünf fünf fünf fünf
sechs sechs sechs sechs sechs sechs
```

```
In [6]: n = input('Bis zu welcher Zahl soll gezählt werden? ')
n = int(n)
```

```
for i in range(1, n + 1):
    wort = ''
    if i == 1:
        wort = 'eins'
    if i == 2:
        wort = 'zwei'
    if i == 3:
        wort = 'drei'
    if i == 4:
        wort = 'vier'
    if i == 5:
        wort = 'fünf'
    if i == 6:
        wort = 'sechs'

    print((wort + ' ') * i)

    if i > n:
        break
```

```
eins
zwei zwei
drei drei drei
```

