**L3\_2\_1 Die for-Schleife (Zählerschleife)**

**Teil 1: Vorbereitung**

# 1.1 Einfache Listen erstellen

In Python kann mit der Anweisung range(*endwert*) eine Liste mit Zahlenwerten angelegt werden. Dabei wird eine Liste erzeugt, die bei 0 beginnt und bis zur letzten Zahl vor *endwert* reicht, also von 0 bis *endwert-1*. Die im folgenden Beispiel mit range(10) erzeugte Zahlenliste beginnt also mit 0 und geht bis 9, nämlich *endwert - 1*, d.h. 10 - 1 = 9:

1 zahlenliste = range(10)

2 # Ausgabe der Liste mit ihren Zahlenwerten:

3 print(list(zahlenliste))

**Datei:** L3\_2\_1\_a\_range-Anweisung.py

Ergebnis des Programms:



Hier sieht man, dass die erzeugte Liste mit 0 beginnt und dann bis zur letzten Zahl vor 10, also bis 9, reicht (denn 9 = 10 – 1).

Analog erzeugt die Anweisung range(5) eine Liste von 0 bis 4:

1 zahlenliste = range(5)

2 # Ausgabe der Liste mit ihren Zahlenwerten:

3 print(list(zahlenliste))

**Datei:** L3\_2\_1\_b\_range-Anweisung.py

Ergebnis des Programms:



# 1.2 Listen mit einem eigenen Startwert erstellen

Man kann der Liste aber auch einen anderen Startwert als 0 geben, dazu wird die Anweisung range(anfangswert, endwert) benutzt:

1 zahlenliste = range(3, 10)

2 # Ausgabe der Liste mit ihren Zahlenwerten:

3 print(list(zahlenliste))

**Datei:** L3\_2\_1\_c\_range-Anweisung.py

Ergebnis des Programms:



# 1.3 Listen mit einer anderen Schrittweite erstellen

Möchte man zusätzlich eine andere Schrittweite verwenden, wird hierfür die Anweisung   
range(anfangswert, endwert, schrittweite) benutzt:

1 zahlenliste = range(1, 16, 2)

2 # Ausgabe der Liste mit ihren Zahlenwerten:

3 print(list(zahlenliste))

**Datei:** L3\_2\_1\_d\_range-Anweisung.py

Ergebnis des Programms:



# 1.4 Aufgaben

1. Ändern Sie das Programm so ab, dass das Programm eine Liste von 0 bis 15 erzeugt.
2. Ändern Sie das Programm so ab, dass das Programm eine Liste von 8 bis 19 erzeugt.
3. Ändern Sie das Programm so ab, dass das Programm eine Liste aller geraden Zahlen von 2 bis 20 erzeugt.
4. Ändern Sie das Programm so ab, dass das Programm eine Liste in absteigender Reihenfolge von 10 bis 1 erzeugt.

Hinweis: Die Lösungen zu diesen Aufgaben finden Sie in der   
Datei L3\_2\_1\_e\_range-Anweisung\_Zusatzbeispiele.py

**Teil 2: Die for-Schleife**

In manchen Fällen muss ein Programm immer wieder die gleichen Anweisungen hintereinander ausführen, bis ein bestimmtes Ergebnis eintritt. Wenn der Programmierer schon vorher weiß, wie oft ein Programm wiederholt werden muss, wird oft die **for-Schleife** benutzt.

Das folgende Programm gibt zeilenweise alle Zahlen von 0 bis 9 aus:

**Beispiel:**

1 for zahl in range(10):

2 print(zahl)

**Datei:** L3\_2\_1\_for\_schleife.py

In Python besteht die for-Schleife aus drei Elementen:

1. Dem Schlüsselwort „for“, mit dem die Schleife eingeleitet wird,
2. einer Variablen (hier: *zahl*), die mit in der Reihe nach alle Werte der folgenden Liste annimmt,
3. einer Liste, in der die Werte stehen, die die angegebene Variable der Reihe nach annimmt. Diese Liste kann mit range(*endwert*) angegeben werden. Nach der Angabe der Liste folgt ein Doppelpunkt als Zeichen dafür, dass hier der Inhalt der Schleife beginnt. In den folgenden Zeilen stehen die Anweisungen, die im Rahmen der Schleife ausgeführt werden sollen, wobei die Zeilen, die zur Schleife gehören, eingerückt sein müssen. Dabei nimmt *zahl* der Reihe nach alle Werte in der Liste an, d.h. im obigen Beispiel nimmt *zahl* im ersten Durchlauf den Wert 0 an, im zweiten Durchlauf den Wert 1, danach 2 und so weiter, bis *zahl* im letzten Durchlauf den letzten Wert der Liste annimmt, also 9.

**Anwendungsbeispiel:**

Es soll ein Programm geschrieben werden, das die 5er-Reihe von 5 bis 50 ausgibt:

1 produkt=0

2 for zahl in range(1, 11):

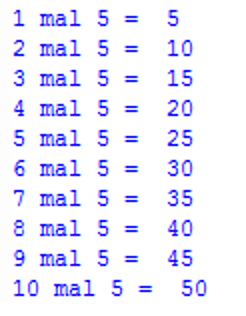
3 produkt = zahl \* 5

4 print(zahl, "mal 5 =", produkt)

**Datei:** L3\_2\_1\_for\_Schleife\_fuenfer\_Reihe.py

In diesem Programm nimmt die Variable *zahl* der Reihe nach die Werte von 1 bis 10 an. Bei jedem Schleifendurchlauf wird *zahl* mit 5 multipliziert und anschließend auf dem Bildschirm ausgegeben, beim ersten Durchgang also 1 mal 5, beim zweiten Durchgang 2 mal 5 etc., bis zum Schluss dann 10 mal 5 berechnet und ausgegeben wird.

**Ausgabe des Programms:**



**Struktogramm:**

