Programmieren mit Python

Teil 3: Schleifen

Dr. Aaron Kunert
aaron.kunert@salemkolleg.de

29. April 2021

Die For-Schleife

Einen Programmabschnitt x-mal ausführen

Problemstellung

Lies eine ganze Zahl `x` ein. Gib dann folgende Zeilen auf der Konsole aus

```
1
2
3
4
⋮
x
```

Wie macht man das?

Lösung

```
x = input("Enter a number")

for k in range(1, x + 1):
  print(k)
```

Struktur der `for...in` Schleife

```
for Variable in range(min, max):
␣␣ Codezeile 1
␣␣ Codezeile 2
␣␣      ⋮
Code, der nicht mehr Teil der Schleife ist
```

Wie funktioniert's?

Die Schleifenvariable wird zunächst gleich dem unteren Wert in `range` gesetzt. Dann wird der `for`-Block wiederholt ausgeführt. Bei jedem Durchgang wird die Schleifenvariable um `1` vergrößert und zwar so lange, wie der Wert der Schleifenvariable kleiner als der obere Wert in `range` ist.

Good to know

- Achtung: Die Schleifenvariable erreicht nie das obere Ende der `range`-Funktion, sondern bleibt immer `1` drunter.
- Die `range`-Funktion ist nicht auf 1er-Schrittweite beschränkt. Mit folgendem Ausdruck werden die Zahlen von `0` bis `9` z.B. in 3er-Schritten durchlaufen: `range(0, 10, 3)`.
- `For`-Schleifen sind flexibel und können alles mögliche durchlaufen, z.B. auch die einzelnen Buchstaben eines Strings (dazu später mehr).

Übung

Einmaleins: Die 7er-Reihe

Schreibe ein kleines Skript, was die 7er-Reihe (bis 70) wie folgt auf der Konsole ausgibt:

```
1 mal 7 ist 7
2 mal 7 ist 14
     ⋮
```

7er-Reihe mit beliebigem oberen Ende

Lies eine positive ganze Zahl `x` ein und gib die 7er-Reihe von 7 bis mindestens `x` wie oben auf der Konsole aus.

Schleife über einen String

Lies Deinen Namen (oder irgendein Wort) auf der Konsole ein und überprüfe, ob er den Buchstaben *a* (groß/klein) enthält.

Schwierigere Übungen

Das Gauss-Problem

Berechne die Summe der Zahlen 1 bis 100.

Fibonacci-Zahlen

Die Zahlenfolge $1, 1, 2, 3, 5, 8, 13 \ldots$ nennt man *Fibonacci*-Folge. Dabei ensteht das Element der Folge, durch die Addition des letzten und vorletzten Elements. Berechne die 30. Fibonacci-Zahl.

Harte Übungen

Quersumme

Lies eine ganze Zahl `x` ein und bestimme ihre Quersumme.

Tipp 1: Die Anzahl der Stellen einer Zahl bekommt man mittels `len(str(x))` heraus.
Tipp 2: Man benötigt Tipp 1 gar nicht.

Zahlenmuster

Gib folgendes Muster auf der Konsole aus:

```
1
1 2
1 2 3
1 2 3 4
    ⋮
1 2 ··· 20
```

Die While-Schleife

Wie die For-Schleife nur abstrakter und open-end

Problemstellung

Lies immer wieder eine Zahl von der Konsole ein. Höre auf, wenn diese Zahl 7 ist.

Wie macht man das?

Lösung

```
x = 0

while x != 7:
  x = input("Enter a number")
  x = int(x)

print("Yeah, you picked the right number.")
```

Struktur der `While`-Schleife

```
while Bedingung:
␣␣ Codezeile 1
␣␣ Codezeile 2
␣␣      ⋮
Code, der nicht mehr Teil der Schleife ist
```

Wie funktioniert's?

Die Schleife wird solange ausgeführt, wie die *Bedingung* `True` ergibt. Nach jedem Durchgang wird der Ausdruck der *Bedingung* neu ausgewertet. Ist die Bedingung `False` wird der Code unterhalb des Schleifenblocks ausgeführt.

Achtung Endlosschleife

Man sollte immer darauf achten, dass die Bedingung in der `while`-Schleife auch wirklich irgendwannmal `False` wird. Ansonsten bleibt das Programm in einer *Endlosschleife* gefangen.

Übung

Ersetze eine `for`-Schleife durch eine `while`-Schleife

Schreib ein Programm, dass alle 7er-Zahlen von 7 bis 700 auf der Konsole ausgibt.

Übung

Notenrechner

Schreib ein Programm, dass wiederholt nach einer Note von Dir fragt und Dir dann jeweils die aktuelle Durchschnittsnote auf der Konsole ausgibt. Das Programm soll durch die Eingabe vom Buchstaben **q** beendet werden können.

Beispielausgabe:

```
Bitte gib eine Note oder q zum Beenden ein: 1
Deine Durchschnittsnote ist 1.0
Bitte gib eine Note oder q zum Beenden ein: 2
Deine Durschnittsnote ist 1.5
    ⋮
```

Übung

Ratespiel

Definiere eine positive ganze Zahl `number_to_guess`. Der User kann nun wiederholt eine Zahl eingeben. Das Spiel endet, wenn die eingegebene Zahl mit `number_to_guess` übereinstimmt. Andernfalls wird auf der Konsole beispielsweise ausgegeben:

```
Sorry, Deine eingegebene Zahl war zu klein, versuche es nochmal:
```

Zusatz 1:
Am Ende soll die Anzahl der Versuche angegeben werden.

Zusatz 2:
Das Spiel soll mit der Eingabe von q abgebrochen werden können.

Zusatz 3:
Google, wie Python die Zahl `number_to_guess` zufällig erzeugen kann (das verbessert das Gameplay).

`break` **und** `continue`

Den Fluss kontrollieren

Das `break`-Statement

Taucht innerhalb einer Schleife das Schlüsselwort `break` auf, so wird die weitere Abarbeitung der Schleife abgebrochen. Die Ausführung wird mit dem Code *nach* dem Schleifenblock ausgeführt.

Beispiel

```
1  for k in range(1,100):
2    print(k)
3    if k > 3:
4      break
5  # prints 1 2 3 4
```

Das `continue`-Statement

Taucht innerhalb einer Schleife das Schlüsselwort `continue` auf, so wird der aktuelle Schleifendurchgang abgebrochen. Die Ausführung wird mit der nächsten Schleifeniteration fortgesetzt.

Beispiel

```
1  for k in range(1,11):
2    if k % 2 == 0:
3      continue
4    print(k)
5  # prints 1 3 5 7 9
```

Der `else`-Block einer Schleife

Analog zum `if`-Statement, kann auch eine Schleife einen `else`-Block haben. Dieser wird ausgeführt, wenn die Schleife *regulär* (also nicht durch die Verwendung von `break`) beendet wird.

Beispiel

```
1  name = input("Your name: ")
2
3  for letter in name:
4    if letter == "a":
5      print("Your name contains an a")
6      break
7  else:
8    print("Your name contains no a")
```

Übung

Die nächste 31er-Zahl

Lies eine ganze Zahl `x` ein. Prüfe, ob eine der 20 auf `x` Zahlen durch 31 teilbar ist. Gib sie in diesem Fall auf der Konsole aus, ansonsten gib aus, dass keine 31er-Zahl gefunden wurde.

Schwierigere Übung

Primzahltest

Lies eine ganze Zahl `x` ein und überprüfe, ob diese Zahl eine Primzahl ist. Das Programm soll etwa folgende Ausgabe liefern

```
Die Zahl 28061983 ist eine Primzahl.
```