Programmieren mit Python

Teil 3: Schleifen

Dr. Aaron Kunert
aaron.kunert@salemkolleg.de
28. Oktober 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 (fast)

```
x = input("Gib eine Zahl ein")
x = int(x)

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

Struktur der `for...in` Schleife

```
for Variable in range(start, end):
␣␣ 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.

Beispiel

```
for x in range(1,5)
  print(2*x)

# prints
# 2
# 4
# 6
# 8
```

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).

Eingangsbeispiel

Schreibe ein Skript, das alle Zahlen von 1 bis 100 auf der Konsole ausgibt.

Lösung

```
for k in range(1, 101)
  print(k)
```

Übungen

Zählen

Zähle auf der Konsole in 7-er Schritten bis 70.

```
7
14
 ⋮
70
```

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
     ⋮
```

Lösungen

Zählen

```
for k in range(1, 11):
  print(7*k)
```

Einmaleins

```
for k in range(1, 11):
  print(f"{k} mal 7 ist {7 * k}")
```

Komplexe Übungen

Zählen in krummen Abständen

Zähle auf der Konsole bis 20, allerdings sollen nur Zahlen ausgegeben werden, die durch 3 oder durch 5 teilbar sind:

3
5
6
9
10
⋮
20

Anzahl bestimmen

Bestimme die Anzahl der Zahlen zwischen 1 und 20, die durch 3 oder durch 5 teilbar sind.

Zählen in krummen Abständen

```
for k in range(1, 21):
  if k % 3 == 0 or k % 5 == 0:
    print(k)
```

Anzahl bestimmen

```
counter = 0
for k in range(1, 21):
  if k % 3 == 0 or k % 5 == 0:
    counter = counter + 1
print(f"Es gibt {counter} gesuchte Zahlen")
```

Das Gauss-Problem

Berechne die Summe der Zahlen 1 bis 100.

Lösung

```
result = 0
for k in range(1, 101):
  result = result + k
print(f"Das Ergebnis ist {result}.")
```

Übung

Schleife über einen String

Lies Deinen Namen auf der Konsole ein und gib die Buchstaben einzeln auf der Konsole auf.

Lösung

```
name = input("Gib Deinen Namen ein: ")

for letter in name:
  print(letter)
```

Übung

Needle-Haystack-Problem

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

Lösung

```
name = input("Gib ein Wort ein: ")

# Flag (Schalter) initialisieren
name_contains_letter_a = False

for letter in name:
  if letter == "a" or letter == "A":
    name_contains_letter_a = True

if name_contains_letter_a:
  print("Der Name enthält ein 'a'.")
else:
  print("Der Name enthält kein 'a'.")
```

Übung mit Trick

Quersumme

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

Tipp: Wandle die Zahl zunächst in einen String um

Lösung

```
number = input("Gib eine Zahl ein: ")
result = 0
# Wir lassen die Zahl als String, damit wir eine Schleife über die Ziffern legen können
for digit in number:
  result = result + int(digit)  # Achtung: digit ist ja eigentlich ein String

print(f"Die Quersumme von {number} ist {result}")
```

Brutale Übung

Fibonacci-Zahlen

Die Zahlenfolge $1, 1, 2, 3, 5, 8, 13 \ldots$ nennt man *Fibonacci*-Folge. Dabei ensteht ein Element der Folge, durch die Addition des vorherigen und vorvorherigen Elements.

Berechne die 30. Fibonacci-Zahl.

Lösung

```python
last = 1  # letzte Zahl
current = 1  # aktuelle Zahl

for k in range(2, 31):
  old_current = current  # Zahl zwischenspeichern
  current = current + last
  last = old_current

print(f"Die {k}-te Fibonacci-Zahl ist {current}")
```

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, solange 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.

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

Schreibe ein Skript, das alle Zahlen von 1 bis 100 auf der Konsole ausgibt. Verwende eine While-Schleife.

Lösung

```
k = 1
while k <= 100:
  print(k)
  k += 1
```

Quizfrage

Schreibe ein Programm, dass solange nach einer Hauptstadt Deiner Wahl fragt, bis die richtige Antwort eingegeben wird.

Beispiel

```
Was ist die Hauptstadt von Frankreich?
```

Darauf hin soll entsprechend der Antwort folgendes Feedback auf der Konsole erscheinen:

```
Das war richtig!
```

bzw.

```
Das war falsch! Versuch's gleich nochmal
```

Lösung

Ratespiel ohne Zusätze

```
answer = input("Was ist die Hauptstadt von Frankreich?")
while answer != "Paris":
  print("Das war leider falsch, versuch es gleich nochmal")
  answer = input("")
print("Das war richtig!")
```

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).

Ratespiel ohne Zusätze

```
number_to_guess = 512
guess = input("Rate meine Zahl: ")
guess = int(guess)

while guess != number_to_guess:
  if guess < number_to_guess:
    print("Deine Zahl war zu klein")
  else:
    print("Deine Zahl war zu groß")
  guess = input("Versuch's nochmal: ")
  guess = int(guess)

print("Du hast gewonnen")
```

Lösung

Ratespiel mit Zusatz 1

```
number_to_guess = 512
guess = input("Rate meine Zahl: ")
guess = int(guess)
counter = 1

while guess != number_to_guess:
  if guess < number_to_guess:
    print("Deine Zahl war zu klein")
  else:
    print("Deine Zahl war zu groß")
  guess = input("Versuch's nochmal: ")
  guess = int(guess)
  counter = counter + 1

print(f"Du hast nach {counter} Versuchen gewonnen")
```

Ratespiel mit Zusatz 2

```
number_to_guess = 512
counter = 1
guess = input("Rate meine Zahl: ")
guess = int(guess)
is_quit = False

while guess != number_to_guess and not is_quit:
  if guess < number_to_guess:
    print("Deine Zahl war zu klein")
  else:
    print("Deine Zahl war zu groß")
  guess = input("Versuch's nochmal: ")
  if guess == "q":
    is_quit = True
  else:
    guess = int(guess)
  counter = counter + 1

if number_to_guess == guess:
  print(f"Du hast nach {counter} Versuchen gewonnen")
```

Lösung

Ratespiel mit Zusatz 3

```python
import random

number_to_guess = random.randint(1, 1000)
guess = input("Rate meine Zahl: ")
guess = int(guess)
# ... usw.
```