Programmieren mit Python

Teil 4: Die While-Schleife

Dr. Aaron Kunert
aaron.kunert@salemkolleg.de

26. Oktober 2022

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("Gib eine Zahl an: ")
  x = int(x)

print("Fertig!")
```

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.

Übung

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
```

Übung

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

Quizfrage

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

Lösung

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")
```

Lösung

Ratespiel mit Zusatz 2

```python
import random

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

Den Schleifenfluss kontrollieren

`break`, `continue` **und** `else`

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

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

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

```
for k in range(1,11):
  if k % 2 == 0:
    continue
  print(k)
# 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

```
name = input("Dein Name: ")

for letter in name:
  if letter == "a" or letter == "A":
    print("Dein Name enthält ein A")
    break
else:
  print("Dein Name enthält kein A")
```

Übungen

Zählen bis zur nächsten 10er-Zahl

Lies eine Zahl `x` ein und gib auf der Konsole die Zahlen von `x` bis zur nächsten 10er-Zahl aus. Ist die Eingabe `x = 17`, so soll die Ausgabe wie folgt aussehen:

```
17
18
19
20
```

Zählen mit Lücken

Schreibe ein Skript, dass die Zahlen von 1 bis 99 aufzählt, dabei allerdings die 10er-Zahlen weglässt. Verwende dabei ein `continue`-Statement.

Zählen bis zur nächsten 10er-Zahl

```
x = input("Gib eine Zahl an: ")
x = int(x)

for k in range(x, x + 11):
  print(k)
  if k % 10 == 0:
    break
```

Zählen mit Lücken

```
for k in range(1, 100):
  if k % 10 == 0:
    continue
  print(k)
```

Quizfrage mit Ausstiegsmöglichkeit

Schreibe ein Programm, dass solange nach einer Hauptstadt Deiner Wahl fragt, bis die richtige Antwort eingegeben wird. Wird allerdings der Buchstabe q eingegeben, so bricht das Programm ab.

Quizfrage mit Ausstiegsmöglichkeit

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

Harte Übung

Primzahltest

Lies eine ganze Zahl `x` ein und überprüfe, ob diese Zahl eine Primzahl ist. Die Ausgabe des Programms soll etwa wie folgt aussehen:

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

Lösung

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

for k in range(2, x):
  if x % k == 0:
    print(f"{x} ist keine Primzahl.")
    break
else:
  print(f"{x} ist eine Primzahl.")
```