Programmieren mit Python

Teil 5: Schleifenfluss, Listen I

Dr. Aaron Kunert
aaron.kunert@salemkolleg.de

11. November 2021

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.

Lösungen

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.")
```

Listen

Viele Variablen gleichzeitig speichern

Problemstellung

Lies mit Hilfe einer Schleife nach und nach Ländernamen ein. Alle Länder sollen dabei gespeichert werden. Danach sollst Du die Möglichkeit haben, das soundsovielte Land anzeigen lassen zu können.

Wie macht man das?

Lösung (fast)

```
# ...
# Um das Eingaben der Länder kümmern wir uns noch
countries = ["Deutschland", "Frankreich", "Italien", "Spanien"]

index = input("Das wievielte Land möchtest Du nocheinmal anschauen?")
index = int(index)

print(f"Das { index }. Land ist { countries[index] }")
```

Struktur einer *Liste*

```
my_list = [element_0, element_1, ..., element_n]
```

Die Variable `my_list` trägt nicht nur einen Wert, sondern $n + 1$ Werte. Ansonsten verhält sich `my_list` wie eine ganz „normale“ Variable. Als Einträge einer Liste sind beliebige Werte mit beliebigen Datentypen zugelassen.

Frage: Welchen Datentyp hat die Liste `[2, 2.3, "Hello"]` ?

Auf Listenelemente zugreifen

Auf das `n`-te Element der Liste `my_list` kann man mittels `my_list[n]` zugreifen.

Mit `my_list[-1]`, `my_list[-2]`, etc. kann man auf das letzte, vorletzte, etc. Element der Liste zugreifen.

Achtung

Python fängt bei 0 an zu zählen. D.h. das erste Element in der Liste hat den Index 0.
Beispiel: `my_list[1]` liefert das **2. Element** der Liste.

Schreibzugriff auf Listenelemente

Nach dem gleichen Prinzip lassen sich einzelne Listeneinträge verändern.
Beispiel: `my_list[3] = "Albanien"`.

Achtung

Man kann nur schon existierende Listeneinträge verändern.

Listeneinträge hinzufügen

Mit der *Methode* `.append()` kann ein Eintrag zur Liste hinzugefügt werden.
Bsp: `my_list.append("Russland")` fügt den String `"Russland"` zu der Liste hinzu.

Listeneinträge entfernen

Mit dem Keyword `del` kann man Einträge an einer bestimmten Position löschen. Dabei verschieben sich die darauffolgenden Einträge um `1` nach vorne.
Beispiel: `del my_list[2]` löscht das dritte Element.

Mit der Methode `.remove()` kann man Einträge mit einem bestimmten Wert löschen.
Beispiel: `my_list.remove("Italien")` entfernt den ersten Eintrag mit dem Wert `"Italien"`. Ist der Wert nicht vorhanden gibt es eine Fehlermeldung.

Übung

Eine Liste erstellen

Schreibe ein kleines Programm, dass Dich ca. 4x nach einem Land fragt, das Du besucht hast und Dir am Ende die Liste der besuchten Länder ausgibt.

Lösung

```
countries = []
for k in range(1, 5):
  country = input("Wo warst Du schonmal im Urlaub? ")
  countries.append(country)
print(countries)
```

Übung

Das Eingangsproblem

Schreibe ein kleines Programm, dass solange Namen von Ländern einliest, bis Du **q** drückst. Danach sollst Du die Möglichkeit haben, eine Zahl `k` einzugeben, so dass das `k`-te Land angezeigt wird.

Lösung

Das Eingangsproblem

```
countries = []
while True:
  country = input("Gib ein Land ein: ")
  if country == "q":
    break
  countries.append(country)

index = input("Das wievielte Land möchtest Du nochmal anschauen?")
index = int(index)
print(f"Das { index }. Land ist { countries[index-1] }.")
```