Programmieren mit Python

Teil 6: Listen II

Dr. Aaron Kunert
aaron.kunert@salemkolleg.de

18. November 2021

Mutability

Listen sind der erste Datentyp, den wir kennenlernen, der *mutable* (veränderbar) ist. Die bisherigen Datentypen waren *immutable*, d.h. man konnte sie zwar überschreiben, aber nicht verändern.

Call by Reference vs. Call by Value

Enthält die Variable `my_list` eine Liste, so speichert Python eigentlich gar nicht die Liste in dieser Variable, sondern nur die Speicheradresse der Liste. Dieses vorgehen nennt man auch *Call by Reference*. Bei den Datentypen `int` und `str` wird stattdessen tatsächlich der Wert der Variable abgespeichert. Dies nennt man *Call by Value*.

Übung

Eine Liste kopieren

Definiere die Variable `my_list` als die Liste `[1,2,3]`. Kopiere die Variable `my_list` in die Variable `my_list_copy`. Füge einen weiteren Eintrag zu `my_list` hinzu. Welchen Wert hat `my_list_copy`?

Lösung

```
my_list = [1, 2, 3]
my_list_copy = my_list
my_list.append(4)
print(my_list_copy) # 1 2 3 4
```

Schleife über Liste

Analog wie über Strings und Ranges kann man Schleifen auch über eine Liste laufen lassen.

Beispiel

```
countries = ["Bulgarien", "Griechenland", "Türkei", "Libanon"]

for country in countries:
  print(country)

# Bulgarien
# Griechenland
# Türkei
# Libanon
```

Schleife über Liste mit Indizes

Möchte man in einer Schleife nicht nur die Listeneinträge, sondern auch die Indizes verwenden, so muss man die Funktion `enumerate()` auf die Liste anwenden.

Beispiel

```
countries = ["Guatemala", "Nicaragua", "Honduras", "Belize"]

for (index, country) in enumerate(countries):
  print(f"Das {index + 1}. Land ist {country}")

# Das 1. Land ist Guatemala
# Das 2. Land ist Nicaragua
# Das 3. Land ist Honduras
# Das 4. Land ist Belize
```

Übung

Liste durchsuchen

Prüfe, ob in einer Liste von Ländern das Land `"Italien"` vorkommt. Gib dazu auf der Konsole entweder

```
Italien ist in der Liste
```

oder

```
Italien ist nicht in der Liste
```

aus.

Lösung

Liste durchsuchen

```
# Wähle ein Beispiel für countries
countries = ["Finnland", "Norwegen", "Schweden", "Dänemark"]

for country in countries:
  if country == "Italien":
    print("Italien ist in der Liste")
    break
else:
  print("Italien ist nicht in der Liste")
```

Ist ein Element in einer Liste enthalten?

Möchte man prüfen, ob ein Element in einer Liste enthalten ist, so kann man auch das Schlüsselwort *in* verwenden.

Beispiel

```
countries = ["Finnland", "Norwegen", "Schweden", "Dänemark"]

var_1 = "Finnland" in countries
var_2 = "Deutschland" in countries

print(var_1)  # True
print(var_2)  # False
```

Eine Liste sortieren

Um eine Liste zu sortieren, verwende die Methode `.sort()`. Dies verändert die Liste dauerhaft.
Um eine sortierte Kopie einer Liste zu erstellen, verwende die Funktion `sorted()`.
Mit Hilfe des Parameters `reverse=True` lässt sich eine Liste absteigend ordnen.

Beispiel für `sort`

```
my_list = [1, 5, 2, 7]
my_list.sort()
print(my_list)  # [1, 2, 5, 7]
```

Beispiel für `sorted`

```
my_list = [1, 5, 2, 7]
sorted_list = sorted(my_list)
print(my_list)  # [1, 5, 2, 7]
print(sorted_list)  # [1, 2, 5, 7]
```

Beispiel für absteigende Sortierung

```
my_list = [1, 5, 2, 7]
my_list.sort(reverse=True)
print(my_list)  # [7, 5, 2, 1]

my_list = [7, 12, 5, 18]
sorted_list = sorted(my_list, reverse=True)
print(sorted_list) # [18, 12, 7, 5]
```

Übung

Beste/Schlechteste Note

Sei `grades` eine Liste der Noten deiner letzten Klausuren (z.B. `grades = [12, 9, 14, 11]`). Gib dann auf der Konsole einmal die beste und einmal die schlechteste Note aus.

Lösung

```
grades = [12, 9, 14, 11]
grades.sort()
min_grade = grades[0]
max_grade = grades[-1]
print(f"Schlechteste Note: {min_grade}")
print(f"Beste Note: {max_grade}")
```

Nützliche Funktionen/Methoden

Für Listen stellt Python viele nützliche Methoden bzw. Funktionen bereit. Wenn Du googlest, findest Du für viele „Alltagsfragen“ eine Lösung.

Zum Beispiel hier: `https://docs.python.org/3/tutorial/datastructures.html`

Beispiele

```
my_list = [2, 4, 8, 1]

len(my_list)    # = 4  (Gibt die Anzahl der Elemente an)
sum(my_list)    # = 15 (Berechnet die Summe der Elemente)
my_list.reverse() # [1, 8, 4, 2] (Dreht die Reihenfolge um)
my_list.insert(2,-1) # [2, 4, -1, 8, 1] (fügt den Wert -1 an Position 2 ein)
my_list.pop() # 1 (Gibt den letzten Eintrag der Liste zurück und entfernt ihn aus der Liste)
```

Übung

Durchschnittsnote

Sei `grades` wieder eine Liste mit deinen letzten Noten. Gib auf der Konsole die Durchschnittsnote aus.

Lösung

```
grades = [12, 9, 14, 11]
total_sum = sum(grades)
count = len(grades)
average = total_sum/count
print(f"Die Durchschnittsnote ist {average}")
```

Slicing

Wenn man eine Liste hat, ist es oft nötig, einen Teil der Liste „auszuschneiden“.
Dafür hat Python die *Slice-Notation* eingeführt.
Diese funktioniert nach folgendem Schema:

`my_list[start:stop:step]`.

Die Einträge (start, stop, step) sind dabei jeweils optional. Wie immer wird der obere Wert (`stop`) gerade nicht erreicht.

Slicing lässt sich übrigens auch nach dem gleichen Schema auch auf Strings anwenden.

Wichtig

Wenn man Slicing anwendet, erhält man eine Kopie der ausgewählten Elemente zurück. Die ursprüngliche Liste wird *nicht* verändert.

Beispiele

```
my_list = [2, 4, 6, 8, 10]

my_list[1:3]      # [4, 6]
my_list[0:4]      # [2, 4, 6, 8]
my_list[1:1]      # []
my_list[0:4:2]    # [2, 6]
my_list[:3]       # [2, 4, 6]
my_list[2:]       # [6, 8, 10]
my_list[:]        # [2, 4, 6, 8, 10]
my_list[1:-2]     # [4, 6]
my_list[-3:-1]    # [6, 8]
my_list[::-1]     # [10, 8, 6, 4, 2]
```