Woche 03: Programmierung – Bearbeiten von Listen in Python

Skript

Erarbeitet von
Ludmila Himmelspach

Lernziele 1
Inhalt 1
Einstieg 1
Zugriff auf Listenelemente 2
Elemente innerhalb einer Liste ersetzen und löschen 5
Take-Home Message 6
Quellen 6
Weiterführendes Material 6
Disclaimer 6

Lernziele

- Auf die Elemente einer Liste zugreifen und diese ausgeben können
- Elemente innerhalb einer Liste ersetzen und löschen können

Inhalt

Einstieg

Du weißt bereits, wie du in Python eine Liste erstellen und dieser neue Elemente hinzufügen kannst. Manchmal muss man aber Elemente in einer Liste bearbeiten, um in dieser Ordnung zu schaffen. In diesem Video lernst du, wie du in Python auf Elemente einer Liste zugreifen

Projekt „KI für alle"
HeiCAD – Heine Center for Artificial Intelligence and Data Science

und diese ausgeben kannst. Außerdem lernst du, wie man Elemente in einer Liste durch andere Elemente ersetzt und löscht.

Zugriff auf Listenelemente

Im Folgenden arbeiten wir mit zwei Listen *kinderbuecher* und *krimis*, die jeweils drei Elemente enthalten.

```
kinderbuecher = ["Peter Pan", "Pippi Langstrumpf", "Alice im Wunderland"]

krimis = ["Mord im Orient-Express", "Alibi", "Der Pate"]
```

Um auf die einzelnen Listenelemente zugreifen zu können, muss man hinter dem Listennamen in den eckigen Klammern den Index des Elements angeben.

Quelle [1]

Wenn wir zum Beispiel wissen wollen, welches Kinderbuch in der Liste *kinderbuecher* an der zweiten Stelle steht, müssen wir hinter dem Listennamen den Index 1 in den eckigen Klammern eingeben.

```
kinderbuecher[1]
```

Wir kombinieren dies direkt mit der *print()*-Anweisung, damit wir die Ausgabe direkt sehen können. An der zweiten Stelle steht also „Pippi Langstrumpf“.

```
print(kinderbuecher[1])
```

```
Pippi Langstrumpf
```

Genauso können wir auch negative Indexe angeben. Das ist zum Beispiel dann praktisch, wenn wir nicht wissen, wie lang unsere Liste ist und wir auf das letzte Element der Liste zugreifen wollen. Denn das letzte Element einer Liste hat den negativen Index -1.

Auf diese Weise können wir das letzte Element der Liste *kinderbuecher* ausgeben.

```
print(kinderbuecher[-1])
```

```
Alice im Wunderland
```

Das ist „Alice im Wunderland“.

Jetzt schauen wir uns noch an, was passiert, wenn wir einen zu hohen Index in der eckigen Klammer angeben. Unsere Liste *kinderbuecher* hat nur drei Elemente, der höchste Index ist also 2. Wir versuchen jetzt das Element mit Index 3 auszugeben. Dafür geben wir Folgendes ein:

```
print(kinderbuecher[3])
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

Cell In[5], line 1

----> 1 print(kinderbuecher[3])

IndexError: list index out of range
```

Da es kein Element an dieser Position in der Liste gibt, bekommen wir eine Fehlermeldung, die besagt, dass unser Index außerhalb des Bereichs liegt. Diese Zeile kommentieren wir direkt wieder aus.

```
#print(kinderbuecher[3])
```

Damit dir solche Fehler nicht passieren, kannst du mit der Funktion *len(),* die für *length* steht, die Anzahl der Elemente einer Liste ausgeben. Das machen wir jetzt für die Liste *kinderbuecher*. Also tippen wir folgende Anweisung ein:

```
print(len(kinderbuecher))
```

```
3
```

Wie erwartet enthält unsere Liste *kinderbuecher* nur drei Elemente.

Wenn wir nur eine Teilmenge aufeinanderfolgender Elemente einer Liste ausgeben wollen, zum Beispiel nur die ersten beiden Elemente, können wir in der eckigen Klammer den Start- und den End-Index getrennt durch einen Doppelpunkt angeben. Dabei wird das Element an der Endposition nicht mit ausgegeben. Diese Methode nennt man Slicing. Wir schneiden so gesehen eine Scheibe aus der Liste aus.

Wir wollen jetzt die ersten beiden Elemente der Liste *kinderbuecher* ausgeben. Dazu müssen wir Folgendes eingeben:

```
print(kinderbuecher[0:2])
```

```
['Peter Pan', 'Pippi Langstrumpf']
```

Wenn wir die Elemente ab einer bestimmten Position bis zum Ende der Liste ausgeben wollen, geben wir nur die Anfangsposition und den Doppelpunkt in den eckigen Klammern an. Mit der Anweisung

```
print(kinderbuecher[1:])
```

geben wir alle Kinderbücher ab der zweiten Position in der Liste aus.

```
['Pippi Langstrumpf', 'Alice im Wunderland']
```

Möchten wir alle Elemente einer Liste von Anfang bis zu einer bestimmten Position ausgeben, müssen wir nur die Endposition nach dem Doppelpunkt in den eckigen Klammern angeben. Mit der Anweisung

```
print(kinderbuecher[:2])
```

werden nur die ersten beiden Kinderbücher ausgegeben.

```
['Peter Pan', 'Pippi Langstrumpf']
```

Um auf die einzelnen Elemente einer Unterliste in einer verschachtelten Liste zugreifen zu können, muss man zwei Indexe angeben. Der erste Index gibt die Position der Unterliste an. Der zweite Index gibt das Element innerhalb der Unterliste an.

Im Folgenden arbeiten wir mit der verschachtelten Liste *buchliste*, die aus zwei Unterlisten besteht.

```
buchliste = [["Pippi Langstrumpf", "Astrid Lindgren", 1945], ["Alice im
Wunderland", "Lewis Carroll", 1865]]

print(buchliste)
```

```
[['Pippi Langstrumpf', 'Astrid Lindgren', 1945], ['Alice im Wunderland',
'Lewis Carroll', 1865]]
```

Um das Erscheinungsjahr vom ersten Buch ausgeben zu lassen, müssen wir hinter dem Listennamen zuerst in den eckigen Klammern den Index 0, weil wir auf das erste Buch zugreifen wollen, und in den zweiten eckigen Klammern Index 2 angeben, weil das Erscheinungsjahr an der dritten Stelle steht.

```
print(buchliste[0][2])
```

```
1945
```

Wenn wir nur einen Index für eine verschachtelte Liste angeben, dann wird die gesamte Unterliste, die an dieser Position steht, ausgegeben.

```
print(buchliste[0])
```

```
['Pippi Langstrumpf', 'Astrid Lindgren', 1945]
```

Elemente innerhalb einer Liste ersetzen und löschen

Wir haben bereits gesehen, dass wir die Listen ändern können, indem wir ihnen neue Elemente hinzufügen. Wir können aber auch Listen verändern, indem wir ihre Elemente ersetzen oder löschen.

Um ein Element der Liste durch ein anderes Element zu ersetzen, müssen wir dieses Element mit Hilfe seines Indexes ansprechen und ihm einen neuen Wert zuweisen. Das schauen wir uns am Beispiel der Liste *krimis* an. Zuerst geben wir die Elemente dieser Liste aus.

```
print(krimis)
```

```
['Mord im Orient-Express', 'Alibi', 'Der Pate']
```

Unsere Liste hat drei Elemente. Nun ersetzen wir den ersten Eintrag der Liste durch einen anderen Krimi. Wir greifen auf dieses Element durch seinen Index zu und ersetzen es durch einen neuen Wert.

```
krimis[0] = "Der dünne Mann"
print(krimis)
```

```
['Der dünne Mann', 'Alibi', 'Der Pate']
```

Wenn wir jetzt die Liste ausgeben, sehen wir, dass das erste Element ausgetauscht wurde. Die übrigen Elemente sind aber gleich geblieben.

Das Löschen der Listenelemente erfolgt mit dem *del*-Befehl, was für *delete* steht. Wenn wir also ein oder mehrere Elemente aus unserer Liste entfernen möchten, müssen wir zuerst den *del*-Befehl eingeben und danach das Element bzw. die Elemente, die gelöscht werden sollen. In unserem Beispiel löschen wir die ersten zwei Elemente. Wenn wir die Liste ausgeben, sehen wir, dass sie nur noch ein Element enthält, das vorher an der dritten Stelle in der Liste gespeichert war.

```
del krimis[0:2]
print(krimis)
```

```
['Der Pate']
```

Wir können auch alle Elemente einer Liste löschen, indem wir den Listennamen gefolgt von einem Punkt und der Methode *clear()* eingeben. Die Liste bleibt noch definiert, aber sie ist dann leer bzw. enthält keine Elemente mehr.

```
krimis.clear()
print(krimis)
```

```
[]
```

Take-Home Message

Mit Listen bietet Python eine mächtige aber eine in der Handhabung einfache Datenstruktur an, die wegen ihrer Flexibilität oft dazu verwendet wird, um komplexe Daten zu verwalten und abzulegen. In diesem Video hast du gelernt, wie man in Python auf die Elemente einer Liste zugreifen und diese ausgeben kann. Außerdem weißt du jetzt, wie man Elemente in einer Liste durch andere Elemente ersetzt und löscht. Damit hast du nur die grundlegenden Funktionen zur Bearbeitung von Listen kennen gelernt. In Wirklichkeit bietet Python noch viel mehr Operationen auf Listen. Wenn du zum Beispiel wissen möchtest, wie man Elemente in einer Liste sortieren oder suchen kann, oder wie man mehrere Listen miteinander verknüpft, schau dir gerne die weiterführende Literatur an. Dort findest du alle wichtigen Methoden und Funktionen dazu.

Quellen

Quelle [1] Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

Weiterführendes Material

Schmitt, S. (2021). *Python Kompendium: Professionell Python Programmieren lernen.* BMU Media Verlag

Barry, P. (2017). *Python von Kopf bis Fuß.* O'Reilly

Disclaimer

Transkript zu dem Video „Woche 03: Programmierung – Bearbeiten von Listen in Python", Ludmila Himmelspach.