

Programmieren mit Python

Teil 7: Dictionaries

Dr. Aaron Kunert

aaron.kunert@salemkolleg.de

25. November 2021

Dictionaries

Problemstellung

Eine Variable soll nicht nur die Namen von Ländern enthalten, sondern auch noch deren Hauptstadt.

Wie macht man das?

Lösung

```
capitals = {"Deutschland": "Berlin", "Spanien": "Madrid", "Italien": "Rom"}
```

```
country = input("Von welchem Land möchtest Du die Hauptstadt wissen?")
```

```
print(f"Die Hauptstadt von { country } ist { capitals[country] } Punkte")
```

Struktur eines *Dictionary*s

```
my_dict = {key_1:value_1, key_2:value_2, ..., key_n:value_n}
```

Das *Dictionary* `my_dict` enthält Schlüssel-Wert-Paare (*key-value-pairs*). Die Schlüssel müssen eindeutig und unveränderlich sein (z.B. vom Typ `string` oder `int`). Die Werte dürfen beliebige Datentypen sein.

Good to know

- Zur besseren Übersichtlichkeit werden Dictionaries oftmals wie folgt formatiert:

```
capitals = {  
    "Deutschland": "Berlin",  
    "Spanien": "Madrid",  
    "Italien": "Rom"  
}
```

- Dictionaries sind mutable, können also verändert werden.
- Dictionaries besitzen keine vernünftige Anordnung und können nicht geordnet werden.
- Ein Dictionary kann leer sein.

- Oftmals bietet es sich an, statt einem Dictionary eine Liste von Dictionaries zu verwenden:

```
countries = [  
    {  
        "name": "Deutschland",  
        "capital": "Berlin",  
        "pop": 82000000,  
        "is_eu_member": True  
    },  
    # ...  
    {  
        "name": "Italien",  
        "capital": "Rom",  
        "pop": 65000000,  
        "is_eu_member": True  
    }  
]
```

Auf Dictionary-Elemente zugreifen

Sei `my_dict = {"a": 5, "b": 8}`.

Mit der Syntax `my_dict["a"]` kann man den Wert an der Stelle "a" auslesen.

Mit der Syntax `my_dict["a"] = 12` kann man einzelne Werte des Dictionaries verändern.

Auf diese Weise können auch ganz neue Paare hinzugefügt werden. Zum Beispiel:

`my_dict["c"] = -2`.

Dictionary manipulieren

Gegeben sei das folgende Dictionary:

```
grades = {"Mathe": 8, "Bio": 11, "Sport": 13}
```

Bestimme die Durchschnittsnote dieser drei Fächer. Verbessere danach Deine Mathenote um einen Punkt und füge noch eine weitere Note für Englisch hinzu (Abfrage über Konsole). Gib danach erneut den Durchschnitt an.

Dictionary manipulieren

```
grades = {"Mathe": 8, "Bio": 11, "Sport": 13}
```

```
grades_sum = grades["Mathe"] + grades["Bio"] + grades["Sport"]
```

```
average = grades_sum/len(grades)
```

```
print(f"Der Durchschnitt ist {average} Punkte")
```

```
grades["Mathe"] += 1
```

```
eng_grade = input("Welche Note hast Du in Englisch? ")
```

```
eng_grade = int(eng_grade)
```

```
grades["Englisch"] = eng_grade
```

```
grades_sum += grades["Englisch"]
```

```
average = grades_sum/len(grades)
```

```
print(f"Der Durchschnitt ist {average} Punkte")
```

Einen Eintrag aus einem Dictionary entfernen

Wie bei Listen, kann man mittels `del`-Statement einen Eintrag aus einem Dictionary entfernen:

```
del eu_countries["united_kingdom"]
```

Was wird hier passieren?

```
old_capitals = {"Deutschland": "Bonn", "Norwegen": "Oslo"}  
new_capitals = old_capitals  
  
new_capitals["Deutschland"] = "Berlin"  
  
print(old_capitals)  
print(new_capitals)
```

Erklärung

Da Dictionaries mutable sind, findet bei ihnen der Aufruf mittels *Call by Reference* statt. Das heißt, dass in der Variable `old_capitals` bzw. `new_capitals` nicht die Länder gespeichert sind, sondern nur die Speicheradresse, wo die Länder zu finden sind. Ändert man die zugrundeliegenden Daten an einer Stelle, so ändern sie sich daher auch an der anderen Stelle.

Eine Kopie von einem Dictionary erstellen

Mit der Funktion `dict()` kann man eine Kopie von einem Dictionary erstellen.

Beispiel: `dict(my_dict)` erstellt eine Kopie von `my_dict`.

Schleife über Dictionary I

Ähnlich wie bei Listen kann man Schleifen auch über ein Dictionary laufen lassen.

Beispiel

```
capitals = {"Litauen": "Vilnius", "Lettland": "Riga", "Estland": "Tallin"}
```

```
for item in capitals:  
    print(item)
```

```
# Litauen
```

```
# Lettland
```

```
# Tallin
```

Schleife über Dictionary II

Möchte man in der Schleife nicht nur die Schlüssel, sondern auch die Werte des Dictionaries zur Verfügung haben, so muss man die Methode `.items()` auf das Dictionary anwenden.

Beispiel

```
capitals = {"Litauen": "Vilnius", "Lettland": "Riga", "Estland": "Tallin"}
```

```
for key, value in capitals.items():  
    print(f"Hauptstadt von {key}: {value}")
```

```
# Hauptstadt von Litauen: Vilnius
```

```
# Hauptstadt von Lettland: Riga
```

```
# Hauptstadt von Estland: Tallin
```

Zwei Dictionaries kombinieren

Gegeben seien zwei Dictionaries, z.B.

```
eu = {"Deutschland": "Berlin", "Frankreich": "Paris" }
```

und

```
non_eu = {"Russland": "Moskau", "China": "Peking" }
```

Füge die Einträge des zweiten Dictionaries zum ersten Dictionary hinzu.

Ein Dictionary „filtern“

Sei ein beliebiges Dictionary mit Noten gegeben. Entferne alle Einträge, deren Note schlechter als 5 Punkte ist.

Zwei Dictionaries kombinieren

```
eu = {"Deutschland": "Berlin", "Frankreich": "Paris" }
non_eu = {"Russland": "Moskau", "China": "Peking" }

for key,value in non_eu.items():
    eu[key] = value
print(eu)
```

Ein Dictionary „filtern“

```
grades = {"Deutsch": 11, "Mathe": 3, "Sport": 14, "Geschichte": 1}
# Man darf die Länge eines Dictionaries in einer Schleife nicht verändern, deshalb machen wir eine Kopie
result = dict(grades)
for key, value in grades.items():
    if value < 5:
        del result[key]
print(result)
```

Ein Dictionary zerlegen

Mit der Methode `.keys()` erhält man eine Liste aller Schlüssel eines Dictionaries.

Mit der Methode `.values()` erhält man eine Liste aller Werte eines Dictionaries.

In beiden Fällen, muss das Ergebnis mittels der Funktion `list()` in eine Liste umgewandelt werden.

Beispiel

```
my_dictionary = {"China": "Peking", "Japan": "Tokio", "Korea": "Seoul"}
```

```
countries = my_dictionary.keys()
countries = list(countries)
```

```
capitals = my_dictionary.values()
capitals = list(capitals)
```

```
print(countries)  # ["China", "Japan", "Korea"]
print(capitals)   # ["Peking", "Tokio", "Seoul"]
```

Comprehensions

Typische Manipulationen

Sehr häufig möchte man eine Datenstruktur (d.h. eine Liste oder ein Dictionary) basierend auf den Werten manipulieren. Dabei werden vor allem zwei Aspekte immer wieder gebraucht: Maps und Filter.

Map

Ersetzt man jedes Element einer Liste durch ein aus dem ursprünglich berechnetem Element, so spricht man von einer *Map* (bzw. einem Mapping).

Beispiel

Gegeben ist die Liste `my_list = [2, 5, 3, 12, 7]`. Die Liste soll so manipuliert werden, dass alle Einträge durch ihren doppelten Wert ersetzt werden.

Traditionelle Lösung

```
my_list = [2, 5, 3, 12, 7]
result = []
for k in my_list:
    result.append(2 * k)
print(result)
```

The Pythonian Way

```
my_list = [2, 5, 3, 12, 7]
result = [2 * k for k in my_list]
print(result)
```

Filter

Streicht man Elemente entsprechend ihres Wertes aus einer Liste, so spricht man von einem *Filter*.

Beispiel

Gegeben ist die Liste `my_list = [2, 5, 3, 12, 7]`. Aus der Liste sollen alle ungeraden Einträge gestrichen werden.

Traditionelle Lösung

```
my_list = [2, 5, 3, 12, 7]
result = []
for k in my_list:
    if k % 2 == 0:
        result.append(k)
print(result)
```

The Pythonian Way

```
my_list = [2, 5, 3, 12, 7]
result = [k for k in my_list if k % 2 == 0]
print(result)
```

Kombination aus Map und Filter

Selbstverständlich können Maps und Filter auch kombiniert werden.

Beispiel

Lösche alle ungeraden Zahlen und verdopple dann alle Zahlen:

```
my_list = [2, 5, 3, 12, 7]
result = [2 * k for k in my_list if k % 2 == 0]
print(result)
```

Dictionary Comprehension

Man kann das gleiche Verfahren auch auf Dictionaries anwenden. Dabei können jeweils key und value für die Maps und Filter verwendet werden.

Beispiel

```
my_dict = {"a": 2, "b": 3}
result = {key: value for (key, value) in my_dict.items()}
print(result)
```

List Comprehension

Gegeben sei eine beliebige Liste von ganzen Zahlen. Streiche alle Zahlen, die ungerade oder negativ sind. Ersetze die übrigen Zahlen durch ihre Hälfte.

Lösung

```
my_list = [2, -3, 40, 15, 7, 8]  
result = [k//2 for k in my_list if k % 2 == 0 and k >= 0]
```

List Comprehension

Gegeben sei das Dictionary `{"Mathe": 9, "Sport": 13, "Physik": 4, "Bio": 12}`. Lösche nun daraus alle Noten unter 5 Punkte sowie die Sportnote. Zusätzlich soll das Dictionary danach wie folgt aussehen: `{"In Mathe": "9 Punkte", "In Bio": "12 Punkte"}`.

Lösung

```
grades = {"Mathe": 9, "Sport": 13, "Physik": 4, "Bio": 12}
result = {
    "In " + key: f"{value} Punkte"
    for (key, value) in grades.items()
    if value >= 5 and key != "Sport"
}
```
