04-01 Dictionaries

November 20, 2020

Empfohlene Unterrichtsinhalte als Vorbereitung für dieses Thema: 02-01/2/3/4/5 (Variablen, Strings, Listen, Integers und Floats, Bool), 03-03 (Schleifen)

1 Einführung in die computerlinguistische Programmierung mit Python

2 04-01: Der Datentyp Dictionary

Dictionarys ermöglichen es uns, Informationen zueinander in Beziehung zu setzen. Zum Beispiel interessiert uns in der Computerlinguistik oft, **wie häufig** bestimmte Wörter/Phrasen/Zeichen/... in einem Text vorkommen.

Dafür erstellen wir ein Dictionary, dessen **Keys** die Strings sind, die uns interessieren, und dessen **Values** die Häufigkeit jedes Strings im Text angeben.

[1]:
```
frequencies = {"und": 12, "oder": 17, "nicht": 5}
print(frequencies)
```

Ab der Pythonversion 3.7 behalten Dictionarys immer die Reihenfolge, in der die Elemente eingefügt wurden. In früheren Pythonversionen können wir uns nicht darauf verlassen, dass das so ist. Das spielt dann eine Rolle, wenn wir z.B. mit einer for-Schleife durch das Dictionary iterieren wollen.

Hier seht ihr, dass Python Dictionarys für identisch hält, wenn...

1. sie die gleichen Keys enthalten
2. und dabei den Keys jeweils die gleichen Werte zugeordnet werden.

[1]:
```
freq1 = {"dog": 3, "cat": 5}
freq2 = {"cat": 5, "dog": 3}
print(freq1 == freq2)
```

Die Informationen, die in einem Dictionary gespeichert sind, können wir abrufen, indem wir den Key angeben, der uns interessiert. Die Rückgabe ist dann der Wert, der zu diesem Key im Dictionary gespeichert ist. Die Syntax dafür ist `<dictionary>[key]`.

[1]:
```
frequencies = {"und": 12, "oder": 17, "nicht": 5}
print(frequencies["und"])
```

Ist ein Key nicht vorhanden, tritt ein Fehler auf:

[1]:
```
frequencies = {"und": 12, "oder": 17, "nicht": 5}
print(frequencies["weil"])
```

Dictionarys sind, genau wie Listen, ein veränderlicher Datentyp (**mutable**). Um einen Wert ins Dictionary einzufügen, schreiben wir:

[1]:
```
frequencies = {"und": 12, "oder": 17, "nicht": 5}

print(frequencies)           # bisheriger Inhalt des Dictionarys
frequencies["weil"] = 15     # dem Schlüssel "weil" soll jetzt der Wert 15␣
 ↪zugeordnet
                             # werden
print(frequencies)           # Dictionary mit zusätzlichem Schlüssel-Wert-Paar
```

Achtung: Jeder Schlüssel kann in einem Dictionary nur genau einmal vorkommen. Es gilt immer der zuletzt definierte Wert für den Schlüssel.

Und schließlich können wir auch durch Dictionarys iterieren. Die `for`-Schleife beginnt fast genauso wie bei Listen. Das aktuelle Element ist immer ein Schlüssel aus dem Dictionary. Wir können also beispielsweise alle Schlüssel-Wert-Paare nacheinander ausgeben:

[1]:
```
frequencies = {"und": 12, "oder": 17, "nicht": 5, "weil": 15}

for word in frequencies:
    print("Wort: " + word)
    print("Häufigkeit: " + str(frequencies[word]))
```

Als Schlüssel für Dictionarys sind nur **unveränderliche Datentypen** erlaubt. Meistens benutzen wir Strings oder Zahlen. Listen sind als Werte möglich, aber nicht als Keys:

[1]:
```
freq3 = {"yes": [1,2,3], "no": [4,5,6]}
print(freq3)
```

[1]:
```
freq4 = { [1,2]: "yes", [4,5]: "no"}      # :(
```

Für das Beispiel, in dem Worthäufigkeiten gezählt werden sollen, brauchen wir Strings für die Wörter und Integers für die Häufigkeiten. Als Schlüssel sollten wir die Wörter wählen.

2.1 Dictionary Methoden

Operation	Auswirkung
list(d)	Gibt eine Liste der Keys eines Dictionarys d aus.
del d[key]	Entfernt einen key aus einem Dictionary d.
d.get(key[, default])	Gibt den Wert des key zurück, oder den default-Wert.
d.items()	Gibt eine Sequenz der Schlüssel-Wert-Paare des Dictionary s d zurück.
d.keys()	Gibt eine Sequenz der Schlüssel zurück.

Operation	Auswirkung
d.values())	Gibt eine Sequenz der Werte des zurück.
d.update(key=value[, key1=value1])	Fügt einem Dictionary neue Schlüssel-Wertpaare hinzu bzw. ändert Werte

Hier wird eine Liste der Keys zurückgegeben

[1]:
```
my_dict = {1: "Alles", 2: "hängt" , 3: "mit", 4: "Allem", 5: "zusammen"}
my_dict_keys = list(my_dict) # Eine Liste der Keys wird erstellt
print(my_dict_keys)
```

Hier wird das Schlüssel-Wert-Paar `5: "zusammen"` entfernt. Das ursprüngliche Dictionary wird verändert.

[1]:
```
my_dict = {1: "Alles", 2: "hängt" , 3: "mit", 4: "Allem", 5: "zusammen"}
del my_dict[5]
print(my_dict)
```

Mit `get` kann man auf den Wert eines Schlüssels zugreifen und einen Default-Wert zuweisen, damit kein Key-Error auftritt. Falls man keinen Default-Wert vergibt und ein Schlüssel nicht existiert wird `None` zurückgegeben.

[1]:
```
my_dict = {1: "Alles", 2: "hängt" , 3: "mit", 4: "Allem", 5: "zusammen"}
wert1 = my_dict.get(5, "Den gibt's nicht")
print(wert1)

print("---")

wert2 = my_dict.get(6, "Den gibt's nicht")
print(wert2)
```

Die Funktionen `d.items()` und `d.keys()` und `d.values()` geben jeweils Sequenzen der Schlüssel-Wert-Paare, der Schlüssel, und der Werte zurück. Diese Sequenzen sind keine Listen. Wir können aus diesen Sequenzen aber leicht Listen erstellen.

[1]:
```
my_dict = {1: "Alles", 2: "hängt" , 3: "mit", 4: "Allem", 5: "zusammen"}
paare = my_dict.items()
schluessel = my_dict.keys()
werte = my_dict.values()
print("---")
for element in [paare, schluessel, werte]:
  print(list(element))
```

Mit `d.update()` lassen sich Werte verändern und Schlüssel-Wert-Paare hinzufügen.

[1]:
```
my_dict = {1: "Alles", 2: "hängt" , 3: "mit", 4: "Allem", 5: "zusammen"}
my_dict.update({6:"oder", 1:"Vieles", 4:"Vielem"})
```

```
print("---")

for schluessel in my_dict:
  print(str(schluessel) + ".\t" + my_dict[schluessel])
```

2.2 Dictionarys Sortieren

Es gibt mehrere Möglichkeiten, die Inhalte eines Dictionarys explizit zu sortieren: Entweder nach der Default-Sortierreihenfolge der Keys, oder nach der Sortierreihenfolge der Values.

- Nach Keys sortiert

[1]:
```
frequencies = {"und": 12, "oder": 17, "nicht": 5, "weil": 15}

print(frequencies)
for word in sorted(frequencies):                    # !!!
    print(word + "\t" + str(frequencies[word]))
```

- nach Values sortiert

[1]:
```
frequencies = {"und": 12, "oder": 17, "nicht": 5, "weil": 15}

print(frequencies)
for word in sorted(frequencies, key=frequencies.get):  # !!!
    print(word + "\t" + str(frequencies[word]))
```

- in umgekehrter Reihenfolge nach Values sortiert

[1]:
```
frequencies = {"und": 12, "oder": 17, "nicht": 5, "weil": 15}

print(frequencies)
for word in sorted(frequencies, key=frequencies.get, reverse=True): # !!!
    print(word + "\t" + str(frequencies[word]))
```

2.3 Zusammenfassung

- Wir haben den strukturierten Datentyp Dictionary kennengelernt, der Sequenzen von geordneten Schlüssel-Wert-Paaren darstellt.
- Dictionaries sind veränderbar. Als Schlüssel dürfen nur unveränderbare Datentypen genutzt werden. Werte können beliebigen Typs sein.
- Mit Hilfe von Dictionary-Methoden können Dictionaries verändert und auf deren Schlüssel und Werte zugegriffen werden.
- Durch die Funktion `sorted(d)` lassen sich Dictionaries sowohl nach Schlüssel als auch nach Werten sortieren.

3 Weitere Themen dieser Woche

- Dateien schreiben
- Dateien lesen