

# Python Dictionaries

# Jacques Mock Schindler

20.11.2024

In Python Dictionaries können ähnlich wie in Python-Listen eine Reihe von Elementen abgelegt werden. Anders als in Python-Listen werden die Elemente allerdings nicht über einen Index aufgerufen, sondern über einen Schlüssel. Man spricht daher von *key - value* Paaren.

## **Erstellen eines Python Dictionary**

Das folgende Listing zeigt ein einfaches Beispiel für ein Python Dictionary.

Die *key - value* Paare werden für die Erstellung eines Python Dictionary in geschweifte Klammern geschrieben. Die Verbindung von Schlüssel und Wert erfolgt durch die Verbindung mit einem Doppelpunkt (*key: value*). Als Schlüssel eignen sich dabei Strings, Tupel (werden noch erklärt), sowie Ganzzahlen. Wichtig ist, dass die Schlüssel aus unveränderlichen Datentypen bestehen.

Um die Lesbarkeit des Codes zu verbessern, hat es sich als gängige Praxis etabliert, die Variabel, welcher das Dictionary zugewiesen wird auf einer separaten Zeile zu schreiben und anschliessend für jedes *key - value* Paar eine neue Zeile zu verwenden. Mehrere Elemente werden durch Kommata abgegrenzt.

## Zugriff auf ein Element in einem Python Dictionary

Um auf einen Eintrag in einem Dictionary zuzugreifen, verwendet man die Variabel mit anschliessenden eckigen Klammern, in denen der Schlüssel steht.

```
1 \NormalTok{definitions[]}\StringTok{\textquotesingle{}\{}list\textquotesingle{}\}}\NormalTok{}}
```

Das obige Listing gibt entsprechend den dem Schlüssel list zugewiesenen Wert aus.

## Direkt über ein Python Dictionary iterieren

Um über die Elemente eines Dictionary zu iterieren gibt es verschiedene Möglichkeiten. Die einfachste Möglichkeit bietet ein *for-loop*.

```
1 \ControlFlowTok{for}\NormalTok{ key }\KeywordTok{in}\NormalTok{ definitions:}
2   \BuiltInTok{print}\NormalTok{(definitions[key])}
```

Diese Möglichkeit hat allerdings den Nachteil, dass nur der Wert und nicht der dazugehörige Schlüssel ausgegeben wird. Um dies zu korrigieren, muss der `print()`-Befehl angepasst werden.

```
1 \ControlFlowTok{for}\NormalTok{ key }\KeywordTok{in}\NormalTok{ definitions:}
2   \BuiltInTok{print}\NormalTok{((key ))}\OperatorTok{+}\StringTok{\\"{}textquotingle\\"{}}: \textquotingle\}
\StringTok{\\"{}textquotingle\\"{}}\}
\OperatorTok{+}\NormalTok{ definitions[key])}
```

Alternativ kann auch ein f-String verwendet werden.

```
1 \ControlFlowTok{for}\NormalTok{\{ key \}\KeywordTok{in}\NormalTok{\{ definitions:}
2   \BuiltInTok{print}\NormalTok{\(\{}\NormalTok{\) \SpecialStringTok{f\textbackslash textquotingle{}\$}
\SpecialCharTok{\{} \NormalTok{\} \NormalTok{\{key\}} \SpecialCharTok{\{} \NormalTok{\} \}\SpecialStringTok{:}
\SpecialCharTok{\{} \NormalTok{\} \NormalTok{\{definitions[key]\}} \SpecialCharTok{\{} \NormalTok{\} \}\SpecialStringTok{`}
\textbackslash StringTok{\textbackslash textquotingle{}\}} \NormalTok{\})}
```

## Über ein Python Dictionary iterieren mit der `.items()` Methode

Eine weitere Möglichkeit bietet eine in die Datenstruktur der Python Dictionaries integrierte Funktion.

```
1 \NormalTok{definitions.items()}
```

Diese Zeile gibt alle Elemente (*items*) des Dictionary aus.

Wenn die Darstellung noch etwas schöner sein soll, kann der Befehl mit einem *for-loop* kombiniert werden.

```
1 \ControlFlowTok{for}\NormalTok{ item } \KeywordTok{in} \NormalTok{definitions.items():}
2   \BuiltInTok{print}\NormalTok{(item)}
```

Um die Elemente nach Schlüssel und Wert getrennt auszugeben, kann der *for-loop* folgendermassen angepasst werden:

```
1 \ControlFlowTok{for}\NormalTok{ item, value } \KeywordTok{in} \NormalTok{definitions.items():}
2   \BuiltInTok{print}\NormalTok{(\text{f}\{key\}\{value\})}
```

Die hier dargelegten theoretischen Ausführungen können in diesem Arbeitsblatt eingeübt werden. Die Musterlösung zum Arbeitsblatt ist hier verlinkt.