

Applied Data Analytics

Basic Python

Dictionaries

Hans-Martin von Gaudecker and Aapo Stenhammar

Contents

- Creating dictionaries
- What can go in a dict
- Accessing elements in dictionaries
- Advantages of labelled data structures

Dictionaries

```
>>> a = {"a": 1, "b": 2, "c": 3}
>>> type(a)
<class 'dict'>
```

```
>>> a["b"]
2
```

```
>>> a["c"] = 42
>>> a
{'a': 1, 'b': 2, 'c': 42}
```

```
>>> a["d"] = 4
{'a': 1, 'b': 2, 'c': 42, 'd': 4}
```

- Map a set of keys to a set of values
- Creation by curly braces and `:` to separate keys and values
- **mutable**: Can add or overwrite entries
- Order is preserved (*since Python 3.6*)

What can go in a dict?

```
>>> nested = {  
>>>     1: {"bla": "blubb"},  
>>>     "two": {"foo": "bar"},  
>>> }
```

- Keys need to be hashable, for example
 - strings
 - ints
 - tuples thereof
- Values can be absolutely anything
- If values are dicts we get nested dictionaries

Accessing elements

```
>>> flat = {"bla": "blubb"}
>>> nested = {
>>>     1: flat,
>>>     "two": {"foo": "bar"}
>>> }
```

```
>>> flat["bla"]
'blubb'
```

```
>>> nested[1]
{'bla': 'blubb'}
```

```
>>> nested[1]["bla"]
'blubb'
```

- Elements are accessed with square brackets
- Chained access for nested dictionaries

When to use dictionaries

- Dictionaries provide label based access
- Lists provide position based access
- Label based access is more readable and less error prone!