Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

FS24 CAS PML - Python

Niklaus Johner

niklausbernhard.johner@bfh.ch

FS24 CAS PML - Python

8. Dictionaries und Sets

Dictionaries

- Ein *dict* enthält *key:item* Paare
- Initialisierung mit Literal:

```
d1 = {}  # Empty dict
d2 = {"key": "value", 3: 4}
```

- Initialisierung mit Klasse:

```
d1 = dict()  # Empty dict
d2 = dict([["key", "value"], (1, 2)])
```

Elementen Zugriff

- Ähnlich zu einer Liste ausser dass
  - Elemente nicht geordnet sind
  - Elemente werden mit ihrem Schlüssel anstatt einem Index ergriffen

```
In [14]: d = {"k": "val", 3: 4}
    ...: print("value for key=k: ", d["k"])
    ...: print("value for key=3: ", d[3])
    ...:
value for key=k:  val
value for key=3:  4
```

Schlüssel

- Schlüssel können fast irgend ein Objekt sein:
  - *int*, *float*, *complex*, *bool*
  - *str*, *tuple*, *function*
- Aber nicht *list, dict* und *set* (weil die veränderlich sind)

```
In [1]: d = {(1, 2): 4, print: "ok", 3.1415: "pi"}
   ...: print("tuple: ", d[(1, 2)])
   ...: print("function: ", d[print])
   ...: print("float: ", d[3.1415])

tuple:  4
function:  ok
float:  pi
```

Elementen zuweisen

- Man kann einem Element einen neuen Wert zuweisen

```
In [35]: d = {"k": "val", 3: 4}
    ...: d["k"] = "new"
    ...: print(d)
    ...:
{'k': 'new', 3: 4}
```

- Ein neues Element zufügen

```
In [36]: d[0] = 1
    ...: print(d)
    ...:
{'k': 'new', 3: 4, 0: 1}
```

Elementen einfügen

- *d1.update(d2)*: Alle *key:value* Paare vom dictionary *d2* in *d1* einfügen
- Ähnlich zu *list.extend*

```
In [38]: d = {"k": "val", 3: 4}
    ...: d.update({"k":"newer", "pi":3.14})
    ...: print(d)
    ...:
{'k': 'newer', 3: 4, 'pi': 3.14}
```

Elementen löschen

- *d.pop(key)*: Das Element mit Schlüssel *key* löschen. Gibt den Wert zurück
- Ähnlich zu *list.pop*

```
In [15]: d = {"k": "val", 3: 4}
    ...: popped = d.pop(3)
    ...: print("popped =", popped,"d =", d)
    ...:
popped = 4 d = {'k': 'val'}
```

keys, values und items

- *d.keys()*: Liste von allen Schlüsseln in *d*

```
In [49]: d={"k": "val", 3: 4}
    ...: d.keys()
    ...:
Out[49]: dict_keys(['k', 3])
```

- *d.values()*: Liste von allen Werten in *d*

```
In [50]: d.values()
Out[50]: dict_values(['val', 4])
```

- *d.items()*: Liste von allen Paaren (Schlüssel, Wert) in *d*

```
In [51]: d.items()
Out[51]: dict_items([('k', 'val'), (3, 4)])
```

Über dictionaries iterieren

- Direkt über den dictionary iterieren, iteriert über die Schlüssel

```
In [53]: d = {"k": "val", 3: 4}
    ...: for k in d:
    ...:     print("key =", k)
    ...:
key = k
key = 3
```

- Über *keys* iterieren

```
In [54]: for k in d.keys():
    ...:     print("key =", k)
    ...:
key = k
key = 3
```

Über dictionaries iterieren

- Über *values* iterieren

```
In [57]: d = {"k": "val", 3: 4}
    ...: for v in d.values():
    ...:     print("val =", v)
    ...:
val = val
val = 4
```

- Über *items* iterieren

```
In [58]: for k, v in d.items():
    ...:     print("k =", k, "val =", v)
    ...:
k = k val = val
k = 3 val = 4
```

Sets

- Ein *set* ist eine ungeordnete Sammlung von eindeutigen und unveränderlichen Objekten
- Initialisierung mit Klasse:

```
s = set()  # empty set
s = set([1, 2, 3, 1])  # set {1, 2, 3}
s = set("letters") # set {'l','e','t','r','s'}
```

- Initialisierung mit Literal:

```
s = {1, 2, 3, 1}  # set {1, 2, 3}
```

Sets: Elemente zufügen und herausnehmen

- s.add(el): Element *el* hinzufügen

```
In [107]: s = {2,3,4}
     ...: s.add(1)
     ...: print("set is",s)
     ...:
set is {1, 2, 3, 4}
```

- s.pop(): Beliebiges Element herausnehmen

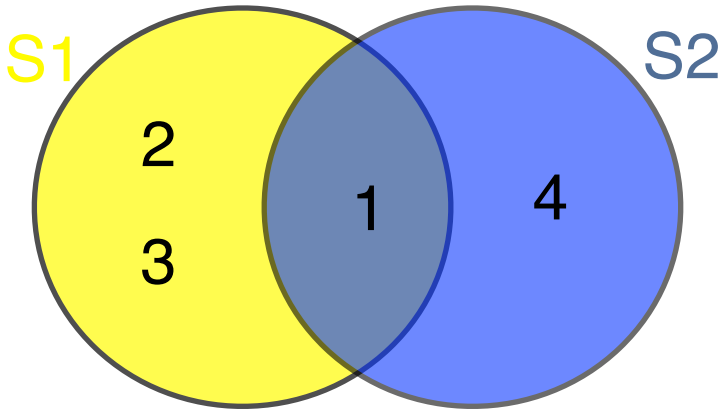
```
In [108]: el=s.pop()
     ...: print("pop=",el)
     ...: print("set is",s)
     ...:
pop= 1
set is {2, 3, 4}
```

- s.remove(el): Element *el* löschen

```
In [109]: s.remove(3)
     ...: print("set is",s)
     ...:
set is {2, 4}
```

Sets: Operationen

- Standard Operationen für Sammlungen:
  - union
  - intersection
  - difference




```
In [116]: s1 = {1, 2, 3}
     ...: s2 = {1, 4}
     ...:

In [117]: s1.union(s2)
Out[117]: {1, 2, 3, 4}

In [118]: s1.intersection(s2)
Out[118]: {1}

In [119]: s1.difference(s2)
Out[119]: {2, 3}
```