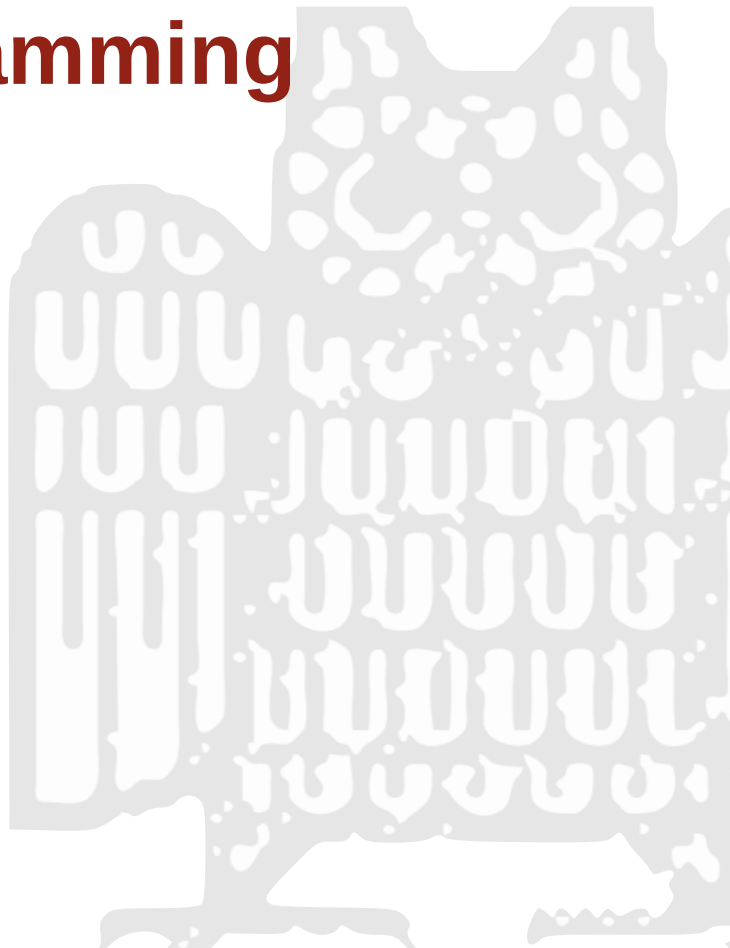
Introduction to Python Programming

09 – Collections: Loose ends

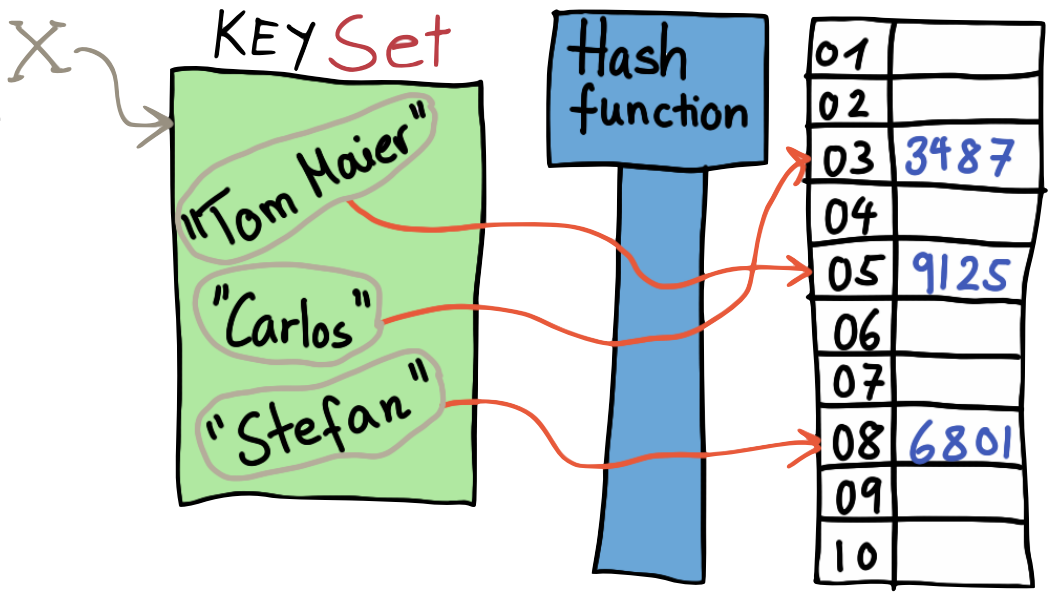
Josef van Genabith (Stefan Thater)
Dept. of Language Science & Technology
Universität des Saarlandes

WS 2022/23



Recap: Dictionaries

- Dictionaries store key-value pairs
- Keys must be unique
- Keys must be hashable
  - all immutable types are hashable




```
myPhoneBook = { "Smith": 3253,
                "Johnson": 3938,
                "Brown": 1443,
                "Miller": 9388 }

print(myPhoneBook["Brown"]) # 1443
```

Recap: Tuples

- Tuples are like lists, but they are **immutable**
  - no append(), no del, no pop(), etc.
- However, tuples can contain mutable values (e.g., lists)
- These values can change!

```
pair = ("Carlos", [1])
pair[1] = [1, 2]  # does not work
pair[1].append(2) # works!
```

- A tuple is hashable if all its items are hashable!

Tuple Notation

```
triple = (1, 2, 3)
tuple = (1, 2)
single = (1,) # why comma?
empty = ()
```

- Tuples vs Lists: Use tuples in cases where you want to make clear that the data will not change
- ... or in cases where you want to use an ordered collection as a key in a dict (or an element of a set).

Sets

- unordered, can contain each element at most once
- may only contain hashable types
  - numbers, strings, booleans, ...
- Empty set: set()
- Achtung: {} creates an empty dictionary, not an empty set!

```
>>> mySet = {5,3,21}
>>> mySet = set([3,5,3,21,4])
>>> mySet
{3,5,21,4}
>>> mySet.add(7)
>>> mySet
{3,5,21,4,7}
>>> mySet.remove(5)
>>> mySet
{3,21,4,7}
```

Sets

- unordered ⇒ one cannot access items via indices
  - s[i] does not work for sets
- Iteraration in the usual way:
  - for item in some_set: ....

Set operations

- Sets support (surprise :-) the usual set operations
  - element: x in s
  - intersection: s1 & s2
  - union: s1 | s2
  - difference: s1 - s2
  - subset: s1.issubset(s2) # read: s1 ⊆ s2
  - ...

frozenset

- sets are mutable
  - all elements must but immutable / hashable
  - the set itself is mutable: we can add and remove items
- frozenset works like a set, but all methods that alter the set (inserting, deleting, changing) are prohibited
- Instantiation:
  - frozen = frozenset([1, 2, 3])

Type Conversion of Collections

- Collections can be easily converted into each other:

```
a = set([1, 2])
b = list(a)
c = tuple(a)
d = tuple(b)
e = set(b)
f = frozenset(e)
```

Type Conversion of Collections

- Collections can be easily converted into each other.
- Dictionaries: only the keys are used
  - my_dict = { "John": 1, "Mary": 2}
  - my_list = list(my_dict) ⇒ ["John", "Mary"]

Summary: Mutability and Hashing

Value	Type	immutable	hashable
17	int	✓	✓
42.0	float	✓	✓
True	bool	✓	✓
"Python"	str	✓	✓
[x,y,z]	list	✗	✗
(x,y,z)	tuple	✓	depends
{x,y,z}	set	✗	✗
frozenset([x,y,z])	frozenset	✓	✓
{"a":x, "b":y}	dict	✗	✗