Theorie / Vorbesprechung Woche 1/2

Variablen

→ keine Angabe des Typs, wird vom interpreter "erkannt"
→ alles ist ein objekt und Variablen sind Referenzen auf Objekte

→ generell kreiert py immer ein neues Objekt
↳ pro Objekt: unveränderlichen Wert
unveränderliche Adresse
↳ Adresse bleibt für die life times eines objekts gleich
↳ id() gibt Adresse zurück

→ Vergleiche mit "==" vs. mit "is"

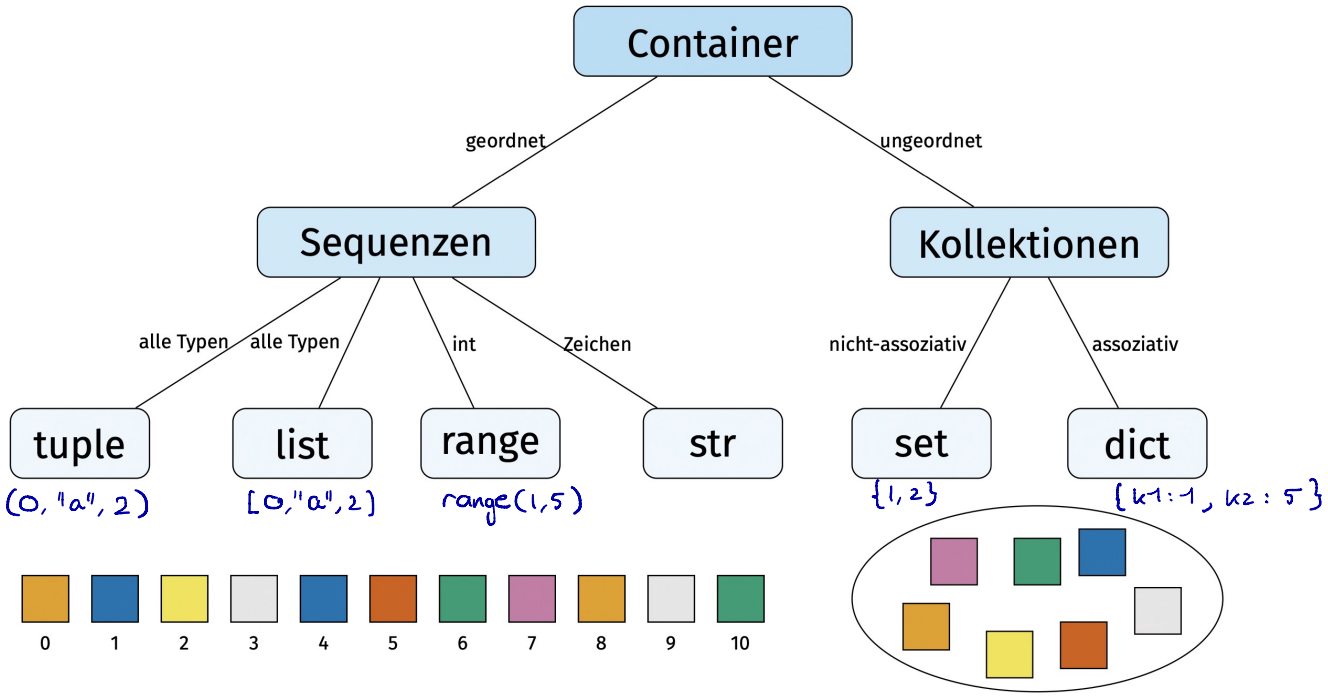
- `==` is for *value equality*. It's used to know if two objects have the same value.
- `is` is for *reference equality*. It's used to know if two references refer (or point) to the same object, i.e if they're identical. Two objects are identical if they have the same memory address.

https://towardsdatascience.com/whats-the-difference-between-is-and-in-python-dc26406c85ad

→ "is" entspricht
id(obj1) == id(obj2)

→ Vorsicht mit Aliasing, da alle Variablen Referenzen sind
↳ .copy() verwenden falls nötig

container



→ verschiedene Arten diese Obj- zu kreieren:

https://www.educative.io/answers/list-vs-tuple-vs-set-vs-dictionary-in-python

→ list ist mutable ≙ veränderlich; tuple, range, str sind immutable

→ wichtige operationen:

↳ zip(a, b)

↳ enumerate(a)

↳ slicing seq[start : stop : stepsize] } startet bei idx start, endet bevor stop

↳ range(start, stop, stepsize) }

↳ a[-1] ≙ a[len(a) − 1] d.h. letztes elem.

list comprehension

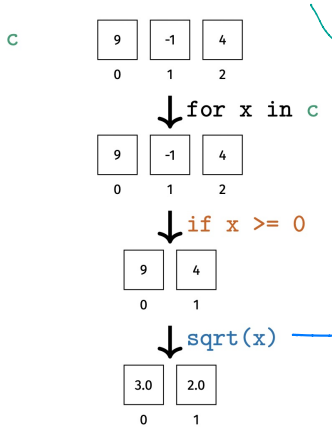
[ ] heisst wir machen hier eine neue Liste

```
[sqrt(x) for x in [9, -1, 4] if x >= 0]
```

Filter

container (hier list)

Aktion/Fkt die wir ausführen wollen



33

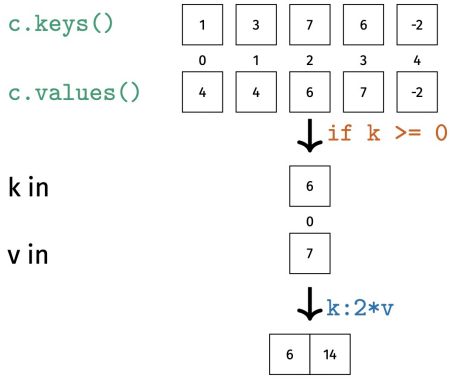
dictionary comprehension

{key: val} heisst wir machen ein neues dict

```
c = {1:4, 3:4, 7:6, 6:7, -2:-2}
{k:2*v for k, v in c.items() if k >= 0 and v % 2 == 1}
```

c

1	4
3	4
7	6
6	7
-2	-2



41

→ Gute TF für dict iteration:

https://realpython.com/iterate-through-dictionary-python/#iterating-through-keys-directly

```
data = {
    'spam':{'amount':12, 'price':0.45},
    'eggs':{'price':0.8},
    'ham':{'amount':5, 'price':1.2}
}

total_prices = {
    food:record['amount']*record['price']
    for food, record in data.items()
    if 'amount' in record
}

# total_prices == {'spam':5.4, 'ham':6.0}
```

keys	values	
'spam'	'amount'	12
	'price'	0.45
'eggs'	'price'	0.8
'ham'	'amount'	5
	'price'	1.2

keys	values
'spam'	5.4
'ham'	6.0

neues dict wird als keys foods haben (was die keys von data sind) und als values record['amount'] * record['price'] haben, wobei record (selbst wieder ein dict) die values von data sind.

→ { bla, bla, bla } → set

{ k : v , bla : bla } → dict