lektion5

November 14, 2019

Table of Contents

1 Identität eines Objekts

2 Listen

2.1 Erzeugen und kopieren

2.2 Flaches und tiefes kopieren (copy / deepcopy)

2.3 Zugriff auf Elemente einer Liste (Slicing von Listen)

2.4 Insert, pop und append

3 Tupel

4 Mengen

4.1 Mengenoperationen

5 Umwandlung

6 Zeichenketten (engl. strings)

6.1 Erzeugen von Zeichenketten

7 Boolsche Variablen und logische Operationen

8 Vergleiche

1 Lektion 5

1.1 Identität eines Objekts

[1]:
```
a=1
b=a
id(a),id(b)
```

[1]:
```
(94577795150624, 94577795150624)
```

[2]:
```
b=2
id(b)
```

[2]: 94577795150656

[3]:
```
id(a)
```

[3]: 94577795150624

1.2 Listen

1.2.1 Erzeugen und kopieren

[4]:
```
leere_liste = []
leere_liste
```

[4]: []

[5]:
```
al = [11,22,33]
```

[6]:
```
bl = al
id(bl), id(al)
```

[6]: (140155637139088, 140155637139088)

[7]:
```
bl[2]= 'a'    # ersetzt in der Liste das Element a[2] durch
bl
```

[7]: [11, 22, 'a']

[8]:
```
al # Achtung
```

[8]: [11, 22, 'a']

[9]:
```
cl = al[:]  # cl ist eine Kopie von al
id(cl)
```

[9]: 140155637561296

[10]:
```
al
```

[10]: [11, 22, 'a']

[11]:
```
cl[0]=23
cl
```

[11]: [23, 22, 'a']

[12]:
```
al
```

[12]:
```
[11, 22, 'a']
```

[13]:
```
l = [1,2,[3,'a']]
bl = l
blc = l[:]
bl[0] = 4
blc[0] = 5
bl[2][0] = 'c'
blc[2][0] ='aa'
```

[14]:
```
l
```

[14]:
```
[4, 2, ['aa', 'a']]
```

[15]:
```
bl
```

[15]:
```
[4, 2, ['aa', 'a']]
```

[16]:
```
blc
```

[16]:
```
[5, 2, ['aa', 'a']]
```

1.2.2 Flaches und tiefes kopieren (copy / deepcopy)

[17]:
```
al = [1,21,[3,4]]
```

[18]:
```
al1 = al.copy()  # flache Kopie
al1[0] = 'hallo'
al1[2][0] = 2225
```

[19]:
```
al
```

[19]:
```
[1, 21, [2225, 4]]
```

[20]:
```
al1
```

[20]:
```
['hallo', 21, [2225, 4]]
```

[21]:
```
from copy import deepcopy
al = [1,21,[3,4]]
```

[22]:
```
al2 = deepcopy(al) # tiefe Kopie
```

[23]:
```
al2[0] = 'Hallo'
al2[2][0] = 5552
```

[24]:
```
al
```

[24]:
```
[1, 21, [3, 4]]
```

[25]:
```
al2
```

[25]:
```
['Hallo', 21, [5552, 4]]
```

1.2.3 Zugriff auf Elemente einer Liste (Slicing von Listen)

[26]:
```
a = [1,2,[3,4],[5,6],7]
a
```

[26]:
```
[1, 2, [3, 4], [5, 6], 7]
```

[27]:
```
a[:-1:] # jedes Element vom Ersten (0) bis Vorletzten (-1)
```

[27]:
```
[1, 2, [3, 4], [5, 6]]
```

[28]:
```
a[0::2]  # jedes zweite Element vom Ersten (0) bis Letzten
```

[28]:
```
[1, [3, 4], 7]
```

[29]:
```
a[0:-1:2]
```

[29]:
```
[1, [3, 4]]
```

1.2.4 Insert, pop und append

[30]:
```
al
```

[30]:
```
[1, 21, [3, 4]]
```

[31]:
```
al.insert(1,6)
```

[32]:
```
al
```

[32]:
```
[1, 6, 21, [3, 4]]
```

[33]:
```
al.pop(2)
```

[33]:
```
21
```

[34]:
```
al
```

[34]:
```
[1, 6, [3, 4]]
```

[35]:
```
al
```

[35]:
```
[1, 6, [3, 4]]
```

[36]:
```
al.append('ende')
al
```

[36]:
```
[1, 6, [3, 4], 'ende']
```

[37]:
```
al + al2  # zusammenfügen von Listen
```

[37]:
```
[1, 6, [3, 4], 'ende', 'Hallo', 21, [5552, 4]]
```

1.3 Tupel

[38]:
```
at = (1,2,4)
bt = at
at, bt
at[1] = 334
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-38-d07675a7cd8a> in <module>
  2 bt = at
  3 at, bt
----> 4 at[1] = 334


TypeError: 'tuple' object does not support item assignment
```

[39]:
```
len(at) # Anzahl der Elemente
```

[39]:
```
3
```

1.4 Mengen

[40]:
```
am = {1,2,3,3}
am
```

[40]: {1, 2, 3}

[41]:
```
len(am)
```

[41]: 3

[42]:
```
am[0]
```

```
␣
↪---------------------------------------------------------------------------
```

```
TypeError                                 Traceback (most recent call last)

<ipython-input-42-88a47290f1cd> in <module>
----> 1 am[0]


TypeError: 'set' object is not subscriptable
```

1.4.1 Mengenoperationen

[43]:
```
am
```

[43]: {1, 2, 3}

[44]:
```
bm = {3,4,5}
bm
```

[44]: {3, 4, 5}

[45]:
```
am | bm  # Vereinigung
```

[45]: {1, 2, 3, 4, 5}

[46]:
```
cm = {1,2}
am < bm  , cm < am # Teilmenge
```

[46]: (False, True)

[47]:
```
am, bm
```

[47]:
```
({1, 2, 3}, {3, 4, 5})
```

[48]:
```
am & bm  # Schnitt
```

[48]:
```
{3}
```

[49]:
```
am - bm # Differenz
```

[49]:
```
{1, 2}
```

[50]:
```
1 in am  # enthalten sein
```

[50]:
```
True
```

1.5 Umwandlung

[51]:
```
aLt = list(at)
aLm = list(am)
aLt, aLm
```

[51]:
```
([1, 2, 4], [1, 2, 3])
```

[52]:
```
aTl = tuple(al)
aTm = tuple(am)
aTl, aTm
```

[52]:
```
((1, 6, [3, 4], 'ende'), (1, 2, 3))
```

[53]:
```
aMt = set(at)
aMl = set(al) # ???
aMt, aMl
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-53-7b3e79cc94ab> in <module>
  1 aMt = set(at)
----> 2 aMl = set(al) # ???
  3 aMt, aMl
```

```
TypeError: unhashable type: 'list'
```

1.6 Zeichenketten (engl. strings)

[54]:
```
text = 'Hallo'
text
```

[54]:
```
'Hallo'
```

[55]:
```
text2 = 'Guten Morgen'
text2
```

[55]:
```
'Guten Morgen'
```

[56]:
```
print(text,text2)
```

```
Hallo Guten Morgen
```

[57]:
```
text2[1::2]
```

[57]:
```
'ue ogn'
```

[58]:
```
eins = str(1)
eins
```

[58]:
```
'1'
```

1.6.1 Erzeugen von Zeichenketten

[59]:
```
from sympy import sin, symbols
x = symbols('x')
f = sin(x)
x1 = 1
'Der Wert von ' + str(f) + ' an der Stelle x= ' + str(x1) + ' ist ' + str(f.
 ↪subs(x,x1).n())
```

[59]:
```
'Der Wert von sin(x) an der Stelle x= 1 ist 0.841470984807897'
```

[60]:
```
# das geht besser mit format
'Der Wert von {0} an der Stelle x = {1} ist {2} '.format(f,x1,f.subs(x,x1).n())
```

[60]:
```
'Der Wert von sin(x) an der Stelle x = 1 ist 0.841470984807897 '
```

[ ]:

1.7 Boolsche Variablen und logische Operationen

[61]:
```
True, False
```

[61]:
```
(True, False)
```

[62]:
```
True and True  # logisches und
```

[62]:
```
True
```

[63]:
```
True & True  # bitweises und
```

[63]:
```
True
```

[64]:
```
False or True # logisches oder
```

[64]:
```
True
```

[65]:
```
False | True # bitweises oder
```

[65]:
```
True
```

[66]:
```
not False  # Negation
```

[66]:
```
True
```

1.8 Vergleiche

[67]:
```
2 <= 2 # kleiner gleich
```

[67]:
```
True
```

[68]:
```
2 <  2 # kleiner
```

[68]:
```
False
```

[69]:
```
1 > 0  # größer
```

[69]:
```
True
```

[70]:
```
1==2, 'a'=='a', (1,2)==(1,2) # gleich
```

[70]:
```
(False, True, True)
```

[71]:
```
 1 != 2  # ungleich
```

[71]: True