

# Organisatorisches

- ▶ Hat es schon jemand geschafft, Python bei sich auf dem Laptop zu installieren?
- ▶ Hat es jemand nicht geschafft und braucht Hilfe?
- ▶ Vorschlag: nach Beendigung der Grundlagen Projektarbeit?  
Wahl zwischen physikalischen, chemischen, ...  
Problemstellungen und anschließende kurze Präsentation

# Listen

## Erstellen einer Liste

```
meine_liste = []  
meine_zweite_liste = list()
```

oder

```
meine_liste = ["a", "b", "c"]
```

oder

```
meine_liste = []  
meine_liste.append("a")  
meine_liste.append("b")  
meine_liste.append("c")
```

## Zugreifen auf ein Element

```
dreier_liste = ["bla", "bli", "Blub"]  
element1 = dreier_liste[0]  
element2 = dreier_liste[1]  
element3 = dreier_liste[-1] # geht auch von hinten
```

## Entfernen von Elementen

```
entferntes_element = meine_liste.pop(2)
```

## Listen miteinander verknüpfen

```
l1 = [1, 2, 3]  
l2 = ["a", "b", "c"]  
liste_gesamt = l1 + l2  
print liste_gesamt
```

## Element in Liste enthalten?

```
buchstaben = ["a", "b", "c", "d", "e"]  
print "c" in buchstaben  
print "f" in buchstaben
```

## Was kann eine Liste speichern?


```
bunt_gemischt = ["hallo", [1, 2, 3], 3.456, 5+3j,  
                 True, False] # geht, muss aber nicht...
```

## Iterieren über Listen

```
for element in bunt_gemischt:  
    print element
```

## Enumerate

```
for i, element in enumerate(bunt_gemischt):
```



## Ein praktisches Beispiel

Ein Chemiker hat in seinem Grundpraktikum alle 20 Sekunden die Konzentration eines bestimmten Stoffes aufgeschrieben. Die resultierende Liste sieht dann so aus:

```
konzentrationen = [1.000,  
                   0.880,  
                   0.756,  
                   0.656,  
                   0.582,  
                   ...  
                   0.130,  
                   0.109,  
                   0.099,  
                   0.079,  
                   0.078,  
                   0.064  
                   ]
```

```
print "Zeit in Sekunden,  Konzentration"  
for index, messwert in enumerate(konzentrationen):  
    print index*20, messwert
```

## Listen “slicen”

- ▶ Möglichkeit auf Teilliste zuzugreifen? (z.B. jedes zweite Element, nur die ersten 9 Elemente, etc.)

```
teil_liste = liste[start:stop:schritt]
```

Bsp: nur jedes dritte Element bis Index 30

Unser Beispiel:

```
teil_liste = zahlenliste[0:30:3]
```

Übungsaufgabe: Teilliste erstellen, *ohne* die Slicing Syntax zu benutzen.

## Merkhilfe für die Index-Benutzung beim Slicen

Folgendes Schaubild (Quelle) macht anschaulich klar, wie die Indizes beim Slicen zu verstehen sind.

```
+---+---+---+---+---+---+
| P | y | t | h | o | n |
+---+---+---+---+---+---+
0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

Trennlinien zwischen den Elementen zählen!



## Listen sortieren

Entweder mit der Methode *sort*, oder mit der Funktion *sorted* sortieren.

### **sorted**

```
l1 = [8, 3, 12, 2]  
l2 = sorted(l1)
```

-> neue sortierte Liste, alte bleibt erhalten!

### **sort**

```
l1 = [8, 3, 12, 2]  
l1.sort()
```

-> l1 wird sortiert, alte Sortierung geht verloren!

## Absteigend sortieren

```
l1 = [8, 3, 12, 2]  
l1.sort(reverse=True)
```

## Unveränderliche Listen (Tupel)

```
mein_tupel = (1, 2, 3)
```

```
meine_liste = [1, 2, 3]
```

```
mein_tupel = tuple(meine_liste)
```

## Listen/Tupel entpacken

```
dreier_tupel = ("Müller", "Daniel", 1.0)
```

Wie speichern wir die drei Werte in die 3 Variablen Vorname, Nachname und Note?

## Methode 1

```
nachname = dreier_tupel[0]  
vorname = dreier_tupel[1]  
note = dreier_tupel[2]
```

## Methode 2

```
(nachname, vorname, note) = dreier_tupel
```

-> linke Seite, rechte Seite brauchen gleich geschachtelte Tupel!

```
name, vorname = "Müller", "Heinz"
```

## Listen zippen (der Reißverschluss)

Wie iteriert man am besten über Elemente mehrerer Listen gleichzeitig?

Beispiel:

```
nachnamen_liste = ["Müller", "Maier", "Schulz"]  
vornamen_liste = ["Daniel", "Dieter", "Elise"]  
noten_liste = [1.0, 2.3, 3.7]
```

```
for vn, nn, note in zip(vornamen_liste, nachnamen_liste, noten_liste):  
    print vn, nn, note
```

Betrachte das Ergebnis von zip:

```
print zip(nachnamen_liste, vornamen_liste, noten_liste)
```

Ausgabe:

```
[('Müller', 'Daniel', 1.0),  
( 'Maier', 'Dieter', 2.3),  
( 'Schulz', 'Elise', 3.7)]
```

-> Liste von Tupeln!

-> In jedem Durchlauf der For-Schleife wird ein Tupel entpackt



## List comprehensions

Bestimme Teilliste aus vorhandener Liste nach bestimmten Kriterien

```
tiere = ["Affe", "Löwe", "Giraffe", "Schlange", "Nashorn"]
```

Nur Tiere mit Namen, die mindestens 5 Buchstaben lang sind:

```
tiere_2 = [tier for tier in tiere if len(tier) >= 5]
```

Ungerade Zahlen von 1 bis 20:

```
ungerade_zahlen = \
[zahl for zahl in range(1, 20) if zahl % 2 != 0]
```

# Strings <-> Listen Vergleich

Gemeinsamkeiten Strings - Listen:

## Addition

```
s1 = "Guten "  
s2 = "Tag"  
s3 = s1 + s2  
print(s3)
```

Ausgabe:

Guten Tag

## Slicen

```
text = "Einen schönen guten Tag"  
print(text[::2]) # gibt jeden zweiten Buchstaben aus
```

Ausgabe:

EnnshnngtnTg

# Mengen (Sets)

Erstellen via

```
menge = set([1, 2, 3, 4, 1, 2])  
print menge
```

oder

```
menge = {1, 2, 3, 4, 1, 2}  
print menge
```

Da eine Menge jedes Element nur einmal enthält, ist die Ausgabe:

{1, 2, 3, 4}

# Mengenoperationen

## Schnitt

```
a = {1, 2, 3, 4}
```

```
b = {1, 4, 6, 7, 8}
```

```
print a & b
```

Ausgabe:

```
{1, 4}
```

## Vereinigung

```
a = {1, 2, 3, 4}
```

```
b = {1, 4, 6, 7, 8}
```

```
print a | b
```

Ausgabe:

```
{1, 2, 3, 4, 6, 7, 8}
```

## Differenz

```
a = {1, 2, 3, 4}
```

```
b = {1, 4, 6, 7, 8}
```

```
print a - b
```

```
print b - a
```

Ausgabe:

```
{2, 3}
```

```
{6, 7, 8}
```

## Symmetrische Differenz

`a = {1, 2, 3, 4}`

`b = {1, 4, 6, 7, 8}`

`print a ^ b`

Ausgabe:

`{2, 3, 6, 7, 8}`



## Test, ob Element in Menge

```
print 1 in {1, 2, 3}  
print "a" in {1, 2, 3}
```

Ausgabe:

True

False

## Set Comprehensions

```
gerade_zahlen = {i for i in range(0, 200, 2)}
```

# Assoziatives Datenfeld bzw. Wörterbücher (Dictionaries)

## Erstellen eines Dictionaries

```
my_dict = dict(a=1, b=2, c=3)
my_dict_2 = {"a":1, "b":2, "c":3}

print my_dict["a"]
```

Ausgabe:

1

## Iterieren über alle Key, Value Paare

```
my_dict = dict(a=1, b=2, c=3)
for key in my_dict:
    print key
```

Ausgabe:

a

c

b

Reihenfolge?

## Iterieren über Key und Value

```
for key, value in my_dict.items():  
    print key, ":", value
```

Ausgabe:

```
a : 1  
c : 3  
b : 2
```

## Dictionary Comprehensions

```
quadrat_zahlen_dict = {i: i*i for i in range(10)}
```