```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#Created on Tue Oct 31 17:38:09 2017
#@author: christianehelzel
```

1 Vektoren und Matrizen mittels 'array'

Listen und Tupel sind fuer numerische Berechnungen ungeeignet. 'np.array' bietet da viele hilfreiche Funktionen.

```
import numpy as np
u = np.array([1,2,3]);
print(u)
print("Der Typ von u: ", type(u))
print("Die Dimension von u:", np.ndim(u))
```

```
[1 2 3]
Der Typ von u:  <class 'numpy.ndarray'>
Die Dimension von u: 1
```

1.1 Nulldimensionale Arrays - skalare Groessen

```
x = np.array(5);
print("Der Typ von x: ", type(x))
print("Die Dimension von x:", np.ndim(x))
```

```
Der Typ von x:  <class 'numpy.ndarray'>
Die Dimension von x: 0
```

1.2 Erzeugung eines Vektors unter Verwendung von 'linspace':

```
v = np.linspace(1,2,4);
print(v)
type(v)
```

```
[ 1.          1.33333333  1.66666667  2.        ]
```

```
numpy.ndarray
```

1.3 Erzeugung eines Vektors unter Verwendung von 'arange'

```
w = np.arange(0,1,0.25)
print(w)
```

```
[ 0.    0.25  0.5   0.75]
```

1.4 Vergleiche mit range-Funktion

```
a = range(1,6);
print(a)
print("Der Typ von a: ", type(a))
print("Die Dimension von a:", np.ndim(a))

b = np.arange(1,6);
print(b)
print("Der Typ von b: ", type(b))
print("Die Dimension von b:", np.ndim(b))
```

```
range(1, 6)
Der Typ von a:  <class 'range'>
Die Dimension von a: 1
[1 2 3 4 5]
Der Typ von b:  <class 'numpy.ndarray'>
Die Dimension von b: 1
```

Mehrdimensionale Arrays (Vektoren, Matrizen)

```
A = np.array([[1,2],[3,4]])
print(type(A))
print(np.ndim(A))
print(A.ndim)
```

```
<class 'numpy.ndarray'>
2
2
```

2 Indizierung / Slicing

Arrays werden (im 2D Fall) zeilenweise gespeichert (wichtig fuer die Indizierung)

```
F = np.array([1,2,3,4,5,6]);
print(F)
print(F[0]) # Das erste Element
print(F[-1]) # Das letzte Element (das kommt VOR Element F[0])
print(F[-2]) # Analog das vorletzte Element
print(F[2:4]) # Das dritte und das vierte Element
print(F[1:5:2]) # Zweites bis fuenftes Element, in Zweierschritten!
print(F[2:]) # Alle ab dem Dritten
print(F[:2]) # Alle bis zum Dritten (exklusive)

B = np.array([[11,12,13],[21,22,23],[31,32,33],[41,42,43]]);
print(B)
print(B[0] [0]) # Von der ersten Zeile das erste Element
print(B[2][1]) # Dritte Zeile und davon der zweite Eintrag
print(B[0,0]) # Kurzschreibweise fuer B[0] [0]
print(B[:,1]) # Alles von der zweiten Spalte

# Einheitsmatrix
np.eye(3)
```

```
np.identity(3)

# Matrix aus Einsen
np.ones((3,3))
# Matrix aus Nullen
np.zeros((3,3))
```

```
[1 2 3 4 5 6]
1
6
5
[3 4]
[2 4]
[3 4 5 6]
[1 2]
[[11 12 13]
 [21 22 23]
 [31 32 33]
 [41 42 43]]
11
32
11
[12 22 32 42]
```

```
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```

2.1 Vorsicht beim kopieren von Arrays!!!

```
x = np.array([[42,22,12],[44,53,66]])
print(x)
y = x
x[0,0] = 1001
print(x)
print(y)
print('')

x = np.array([[42,22,12],[44,53,66]])
y = x.copy()
x[0,0] = 1001

print(x)
print(y)
```

```
[[42 22 12]
 [44 53 66]]
[[1001   22   12]
 [  44   53   66]]
[[1001   22   12]
 [  44   53   66]]

[[1001   22   12]
 [  44   53   66]]
[[42 22 12]
 [44 53 66]]
```

Im ersten Fall zeigen ‘x’ und ‘y’ beide auf den selben Speicherbereich. Deshalb wird auch ‘y’ geaendert wenn man in ‘x’ einen Eintrag aendert.

2.2 Kopieren einer Teilmatrix

Das gleiche geht auch mit Teilmatrizen

```
x = np.array([[42,22,12],[44,53,66]])
y = x[:,0].copy()
x[0,0] = 1001

print(x)
print(y)
```

```
[[1001   22   12]
 [  44   53   66]]
[42 44]
```

Weitere Informationen zur Arrayerzeugung und zum Slicing finden Sie unter anderem auf:
https://www.python-kurs.eu/numpy_arrays_erzeugen.php