Übung 2

KUNAL OBEROI UND MATTHIAS TSCHÖPE

FACHBEREICH: INFORMATIK

VORLESUNG: KÜNSTLICHE INTELLIGENZ

Agenda

- Numpy
- Vektoren Operationen Allgemein
- Matrix Operationen Allgemein
- Definieren von Arrays
- Lineare Algebra mit Numpy , Arrays mit Nullen und Einsen
- Dimension, Shape und Axis
- Eingebaute und Akkumulierte Funktionen
- Zugriff auf Zeilen und Spalten
- Daten mit Numpy einlesen
- Matplotlib
- Übungaufgaben

Numpy

- Fundamentales Packet für numerische Operationen mit Python
- N-dimensionales Array Objekt
- Schnelle mathematische Operationen über Arrays möglich z.B zeros, ones
- Lineare Algebra, Fourier Transformation, Generation von Zahlen
- Effiziente Schnittstelle zum Speichern und Verarbeiten dichter Daten
  - pip install numpy
  - import numpy as np

Vektoren Operationen Allgemein

Vektoraddition	Skalarprodukt

$$\vec{a}+\vec{b}=\begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix}+\begin{pmatrix}b_1\\b_2\\b_3\end{pmatrix}=\begin{pmatrix}a_1+b_1\\a_2+b_2\\a_3+b_3\end{pmatrix}.$$

$$\vec{a}\cdot\vec{b}=\begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix}\cdot\begin{pmatrix}b_1\\b_2\\b_3\end{pmatrix}=a_1b_1+a_2b_2+a_3b_3,$$

Matrizen Operationen Allgemein

Matrixaddition	Multiplizieren

$$\mathbf{A}+\mathbf{B}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

$$R^{l\times m} \times R^{m\times n} \to R^{l\times n}, \quad (A,B) \mapsto C = A \cdot B$$

$$c_{ik} = \sum_{j=1}^{m} a_{ij} \cdot b_{jk},$$

$$A \cdot B = \begin{pmatrix} 1 & -3 \\ 0 & -2 \end{pmatrix} \cdot \begin{pmatrix} -1 & 4 \\ 7 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cdot(-1) + (-3) \cdot 7 & 1 \cdot 4 + (-3) \cdot 1 \\ 0 \cdot (-1) + (-2) \cdot 7 & 0 \cdot 4 + (-2) \cdot 1 \end{pmatrix}$$

$$= \begin{pmatrix} -22 & 1 \\ -14 & -2 \end{pmatrix}.$$

Definieren von Arrays

```
import numpy as np
```

Bsp:

```
>>> A = np.array([[1,2,3],[4,5,6]])
>>>print(A)
```

Output:

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

Lineare Algebra mit Numpy

Vektoren

```
In [1]:
1  import numpy as np
2  #Dot product
3
4  A = np.array([3,5,2])
5  B = np.array([5,3,2])
6

In [2]:
1  C = np.vdot(A,B)

In [3]:
1  print(C)

34
```

Matrizen

```
In [16]:
#Matrix Multiplikation
A = np.array([[3,5,2],[1,1,2],[2,7,4]])
B = np.array([[5,3,2],[2,1,1],[4,7,2]])

In [17]:
C = np.matmul(A,B)

In [18]:
print(C)

[[33 28 15]
 [15 18  7]
 [40 41 19]]
```

https://docs.scipy.org/doc/numpy/reference/routines.linalg.html#matrix-and-vector-products

Array mit Nullen und Einsen

Array mit Länge 10, gefüllt mit Integer Werten

```
>>> np.zeros(10)
Output: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

3x5 Array mit Integer Werten

```
>>> np.ones((3,5),dtype=np.int32))
Output: array([[1, 1, 1, 1, 1],
               [1, 1, 1, 1, 1],
               [1, 1, 1, 1, 1]])
```

Dimension, Shape , Axis

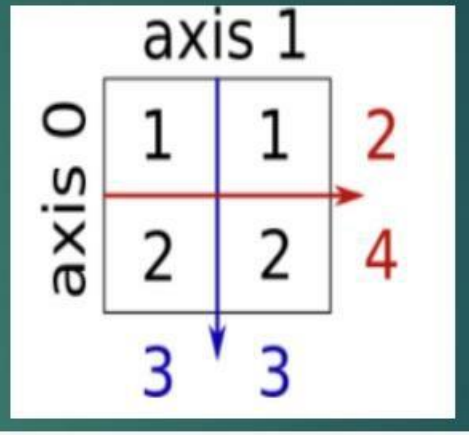
```
In [39]: a = np.arange(10).reshape((2,5))

In [40]: a
Out[40]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])

In [41]: a.ndim
Out[41]: 2

In [42]: a.shape
Out[42]: (2, 5)

In [43]: a.size
Out[43]: 10
```




```
In [2]: import numpy as np
        x = np.arange(12).reshape((3,4))

In [3]: x
Out[3]: array([[ 0,  1,  2,  3],
               [ 4,  5,  6,  7],
               [ 8,  9, 10, 11]])

In [4]: x.sum(axis=1)
Out[4]: array([ 6, 22, 38])
```

Eingebaute und Akkumulierte Funktionen

```
In [23]: x = [0, np.pi/6,np.pi/4,np.pi/3] #pi/6, pi/4, pi/3 in Bogenmaß

In [25]: y = np.sin(x)

In [26]: y #Sinus-Werte
Out[26]: array([0.        , 0.5       , 0.70710678, 0.8660254 ])

In [27]: z = np.cos(x)

In [28]: z #Kosinus-Werte
Out[28]: array([1.        , 0.8660254 , 0.70710678, 0.5       ])
```

```
In [47]: #Akkumulierte Funktionen

         #Max,Mean,Sum

         x = np.arange(12).reshape(4,3)

In [48]: x
Out[48]: array([[ 0,  1,  2],
                [ 3,  4,  5],
                [ 6,  7,  8],
                [ 9, 10, 11]])

In [55]: x.max(axis=1)
Out[55]: array([ 2,  5,  8, 11])
```

```
In [60]: y = np.array([[4,2,1,3],[50,20,10,30]])
         y

Out[60]: array([[ 4,  2,  1,  3],
                [50, 20, 10, 30]])

In [62]: y.mean(axis=0)

Out[62]: array([27. , 11. ,  5.5, 16.5])
```

Zugriff auf Zeilen, Spalten und Zellen


```
In [64]: x = np.array([1,2,3,4,5,6])

In [65]: x[0]

Out[65]: 1

In [66]: x[-2]

Out[66]: 5

In [67]: x[-1]

Out[67]: 6
```

```
In [69]: x = np.arange(4).reshape((2,2))

In [70]: x

Out[70]: array([[0, 1],
                [2, 3]])

In [71]: x[0,0]

Out[71]: 0

In [72]: x[0,1]

Out[72]: 1

In [74]: x[1,1]

Out[74]: 3

In [75]: x[1,0]

Out[75]: 2
```

```
In [12]: x = np.array([0,5,20,2,1,3,8,9,10,20,20,30,40])

In [13]: print(x[1:7:2])

         [5 2 3]
```

```
In [14]: x = np.array([[[1],[2],[3]], [[4],[5],[6]]])

In [16]: x[:,0]

Out[16]: array([[1],
                [4]])
```

```
In [2]: import numpy as np


        A = np.array([[1,2,3],[3,5,8]])

In [28]: A[0:,0]

Out[28]: array([1, 3])
```

Daten mit Numpy einlesen

- Hinweis: Die Datei der Daten muss entweder im gleichen Ordner liegen oder ihr gibt den Pfad zur Datei an!

```
In [6]: #Datei einlesen und Werte ausgeben
        import numpy as np

        x_data = np.loadtxt("TemperatureData.out")

In [8]: x_data

Out[8]: array([27.25, 27.  , 27.25, ..., 26.75, 27.25, 27.  ])
```

Matplotlib

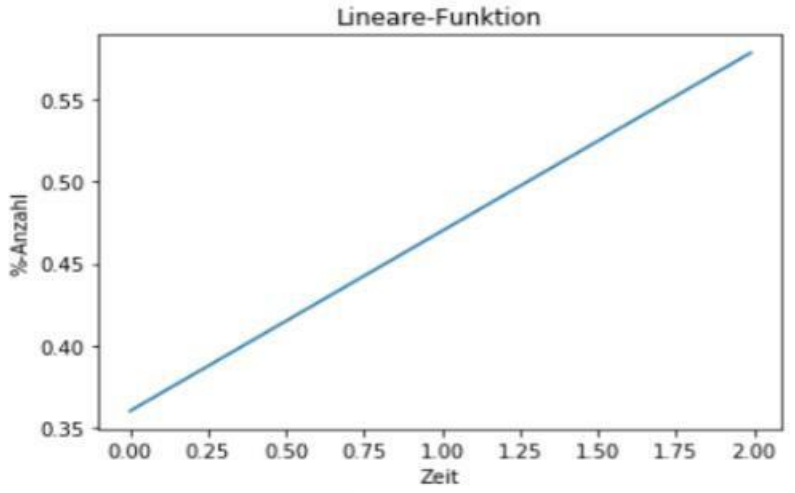
- Linie- Plotten

```
In [32]: import matplotlib
         import matplotlib.pyplot as plt
         import numpy as np

         #y = mx + b
         m = 0.11
         b = 0.36
         x = np.arange(0.0, 2.0, 0.01)
         y = m*x + b


         plt.plot(x,y)
         plt.title("Lineare-Funktion")
         plt.xlabel("Zeit")
         plt.ylabel("%-Anzahl")


         plt.show()
```



Linien Operationen

- Farbe ändern, Breite und Style

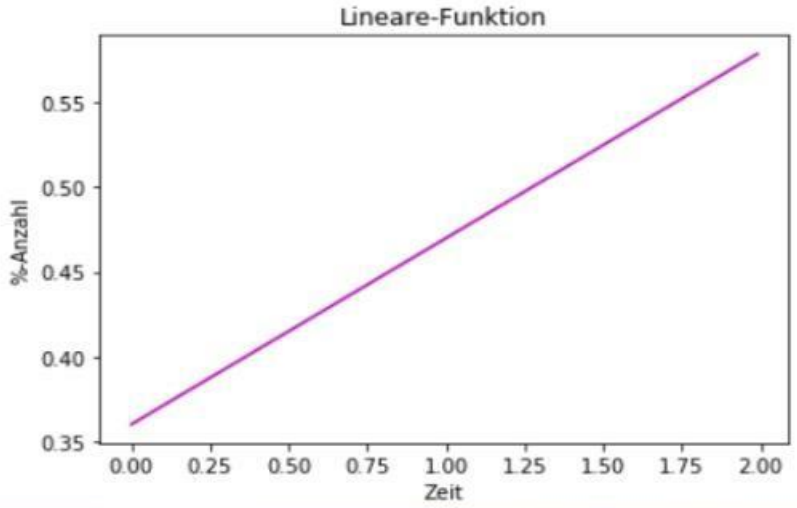
b	blau	**m**	magenta
g	grün	**y**	gelb
r	rot	**k**	schwarz
c	türkis	**w**	weiß

```
In [33]: import matplotlib
         import matplotlib.pyplot as plt
         import numpy as np

         #y = mx + b
         m = 0.11
         b = 0.36
         x = np.arange(0.0, 2.0, 0.01)
         y = m*x + b


         plt.plot(x,y,c = 'm')
         plt.title("Lineare-Funktion")
         plt.xlabel("Zeit")
         plt.ylabel("%-Anzahl")


         plt.show()
```



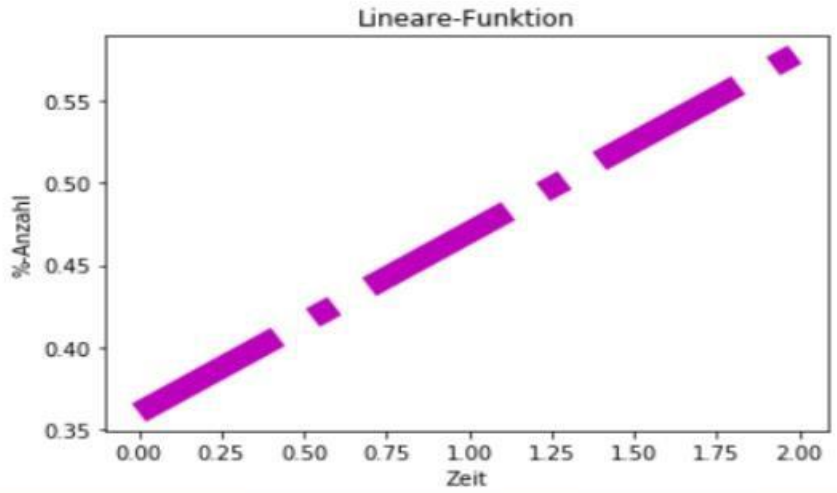
In [41]:

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

#y = mx + b
m = 0.11
b = 0.36
x = np.arange(0.0, 2.0, 0.01)
y = m*x + b


plt.plot(x,y,"-.",c = 'm',linewidth = 12,markersize=10)
plt.title("Lineare-Funktion")
plt.xlabel("Zeit")
plt.ylabel("%-Anzahl")


plt.show()
```



Übungsaufgaben Aufgabe1

- Lösen Sie das Gleichungssystem mit Hilfe von Numpy.
- Berechnen Sie die Matrixgleichung

$$3x + 6y - 5z = 12$$
$$x - 3y + 2z = -2$$
$$5x - y + 4z = 10$$

$$\begin{bmatrix} 3 & 6 & -5 \\ 1 & -3 & 2 \\ 5 & -1 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 12 \\ -2 \\ 10 \end{bmatrix}$$

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}.$$

Übungsaufgaben

Aufgabe 2

- 1. Erstellen sie eine Liste 0 bis 10 mit Schrittgröße 0.01.
- Geben sie die Shape, Dimension und die Summe von Axis 0 aus.
- Interpretieren sie diese Lösung!
- 2. Erstellen sie ein 5x5 Array mit Zufallszahlen und bestimmen sie das Maximum und Minimum
- 3. Erstellen Sie ein 2D-Arraymit 1 am Rand und 0 innerhalb
- 4. Definieren sie ein Array im Intervall [1,17] mit der Shape 4x4, berechnen sie die Transponierte sowie die Inverse Matrix
- Hinweis: Verwenden sie die Numpy Funktion arange()

Aufgabe 3

- 1. Erstellen sie ein Numpy Array mit 5000 gleichverteilten Zufallszahlen
- 2. Erstellen sie ein Numpy Array mit 5000 normalverteilten Zufallszahlen
- 3. Berechnen sie den Mittelwert und die Standardabweichung
- 4. Histogramme und Scatter visualisieren mit den gleichverteilten und normalverteilten Zufallszahlen
- Hinweis: Für Die Berechnung der Mittelwert und Standardabweichung gibt's in numpy bereits vorimplementierte Funktionen

Aufgabe 4

Die Temperatur eines Objektes wurde mit einer thermischen Kamera von einer Abstand von 2 Metern gemessen. Die Temperatur während der Messung war konstant. Die Liste der Messwerte befindet sich in der Datei "TemperaturDaten.out".

1. Datei einlesen und Werte ausgeben. Hinweis: Mit Numpy können einfach CSV Dateien eingelesen und auch gespeichert werden. Siehe loadtxt(…) und savetxt(..)
2. Um die Verteilung der Messwerte zu sehen, sollen sie auf einem Histogramm dargestellt werden, siehe Aufgabe 3.
3. Berechnen Sie den Mittelwert und die Standardabweichung
4. Berechnen Sie für diesen konkreten Fall, wie viel Prozent der Messwerte innerhalb der Intervalle $[\mu \pm \sigma]$, $[\mu \pm 2*\sigma]$ bzw. $[\mu \pm 3*\sigma]$ liegt. Hinweis: Eine Funktion kann hilfreich sein beim lösen dieser Aufgabe