

## Data Science

### Übung 7

#### Einführung in die Programmierung

##### Über dieses Übungsblatt

In diesem Übungsblatt geht es darum erste Schritte im Bereich Data Science zu tun: wir werden Daten aus Textdateien einlesen, verarbeiten und anzeigen. Einfach zu benutzende aber mächtige Bibliotheken wie [Numpy](#) oder [Matplotlib](#) sind der Grund warum Python für Data Science so beliebt ist. In diesem Blatt möchten einerseits verstehen wie diese Bibliotheken funktionieren, aber auch lernen sie zu benutzen. Deshalb sind externe Bibliotheken nur erlaubt, falls es explizit im Aufgabenteil dabeisteht. Sie können wie gewohnt ihre Bibliotheken mit dem PyPy Paketmanager über den Konsolenbefehl `pip install matplotlib` bzw. `pip install numpy` installieren.

## Aufgabe Numpy / Slicing

*Letzte Änderung: 15. June 2023, 12:38 Uhr*

16 Punkte — [im Detail](#)

Ansicht:  | 

In dieser Aufgabe geht es darum, einige Features von Python's wahrscheinlich beliebtesten Bibliothek [Numpy](#) kennenzulernen und Python's sehr nützliche Slicing-Notation zu verstehen. Numpy enthält einen sog. [Wrapper](#) für Vektoren und (verallgemeinerte) Matrizen, die `array` genannt werden. Anstatt Listen von Zahlen zu verwenden, benutzen wir also Numpy Arrays. Dies hat viele Vorteile:

- deutlich schnellere Laufzeiten als naiver Python-Code
- viele Operationen (Vektoraddition, Skalarmultiplikation, Matrizenmultiplikation etc.) sind schon definiert und funktionieren einfach mit den Python-Operatoren `+`, `*`, `-`, `/` etc.
- viele häufig gebrauchte Methoden gibt es schon (Eigenwerte von Matrizen bestimmen, sortieren eines Arrays etc.)
- die Notation ist intuitiv und einfach

Auch wenn in diesem Kurs eher Programmiersprachen übergreifend gültige Denkart und Strukturen unterrichtet werden sollen, widmen wir Numpy trotzdem eine Aufgabe, da es wahrscheinlich das wichtigste Werkzeug in Python ist.

**Hinweis:** Sie finden ein Kurzsript zur Verwendung von Numpy [hier](#)

**Hinweis2:** Verwenden Sie für diese Aufgabe **keine** Schleifen.

0) Importieren Sie Numpy mit dem Befehl: `import numpy as np`. Typischerweise schreibt man alle `import`-Befehle an den Anfang des Programmes. Im Anschluss kann man alle Funktionen der `numpy`-Bibliothek mit `np.[methodenname]([argumente])` verwenden. Nun muss man nur die richtige Funktion für das Problem das man hat finden; dies geschieht typischerweise über einfaches Suchen von "[was möchte ich tun] numpy" auf der Suchmaschine ihrer Wahl oder lesen der [Numpy Dokumentation](#). Lesen Sie sich die Dokumentation oder IDE-Kurzbeschreibung der Funktionen kurz durch, die wir im Anschluss verwenden werden, um zu verstehen wie diese funktionieren.

Erstellen Sie ein 1D-Array (also eine Liste mit mehr Features) mit den Zahlen von 42-420 mit der `np.arange` Funktion.

**1a)** Bilden Sie das Produkt und Summe der Zahlen in dem Array. Verwenden Sie dazu die `np.sum` und `np.prod` Funktionen. Überlegen Sie sich wieso das Ergebnis des Produkts einen unerwarteten Wert zurückgibt.

**1b)** Berechnen Sie die Summe der ersten 18 Zahlen im Array mithilfe der Slicing-Notation `array[:x]`.

**1c)** Berechnen Sie die Summe über jede zweite Zahl im Array, angefangen bei der nullten Zahl mithilfe der Slicing-Notation `array[::x]`.

**1d)** Berechnen Sie die Summe der letzten 18 Zahlen im Array mithilfe von negativer Indizierung `array[-x]` und der Slicing-Notation `array[x:]`.

**1e)** Erstellen Sie eine boolsche Maske `mask` aller ungeraden Zahlen (`x` ist gerade gdw. `x % 2 == 0` und ungerade gdw. `x % 2 == 1`) im Array. Indizieren Sie das Array mit der Maske (ja, das geht!) mit `array[mask]` und beobachten Sie wie nur die Zahlen übrig bleiben, die ein `True` in der Maske besitzen. Man kann so beispielsweise auch eine Maske aller positiven Werte erstellen `array > 0`.

Erstellen Sie ein zweidimensionales Array (also eine Matrix) bestehend aus zufälligen `int`-Werten in `[0,10]` mit Dimensionen `20x30`, mit der `randint` Methode von Numpy.

**2a)** Zählen Sie, wieviele Werte in den ersten beiden Spalten des Arrays größer oder gleich zwei sind. **Tipp:** Man kann die `sum` Funktion auf eine bool'sche Maske anwenden und es werden die Anzahl an `True`-Werten gezählt.


**2b)** Setzen Sie alle Werte an den "Rändern" (also erste und letzte Zeile/Spalte des Arrays) auf -1.

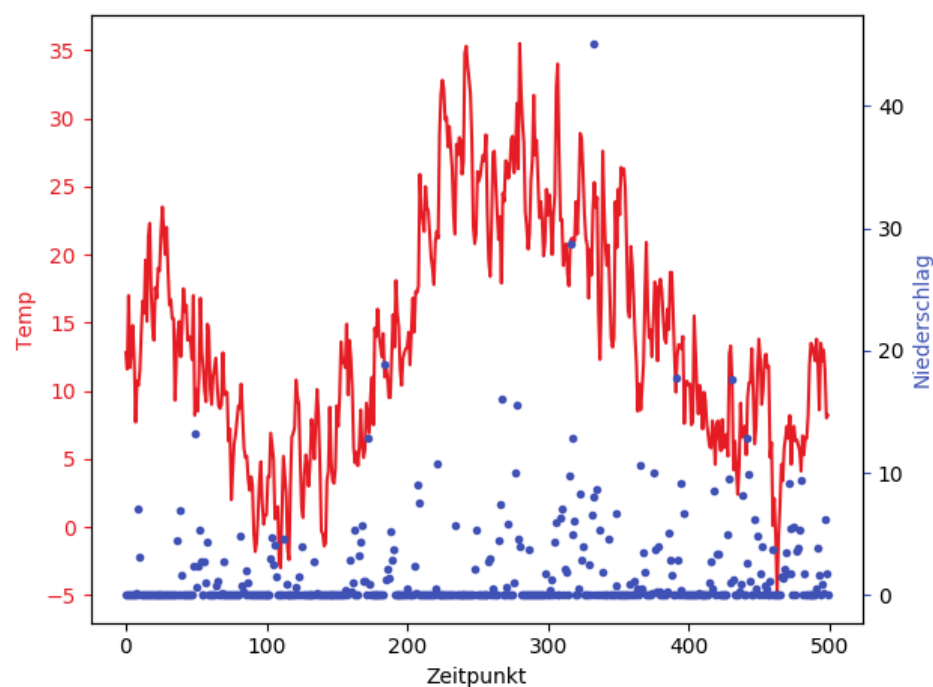
**2c)** Setzen Sie einen Wert an einer zufälligen Zeile (ausgewählt mit der `randint` Methode) in der letzten Spalte auf -2.

## Aufgabe Matplotlib

*Letzte Änderung: 15. June 2023, 12:38 Uhr*

**24 Punkte** — [im Detail](#)

Ansicht:  | 



Wetterdaten für Frankfurt am Main.

In dieser Aufgabe geht es um die einfache Darstellung von Daten, die aus einer Textdatei eingelesen werden. Dies ist einer der Stärken von Python, da das Einlesen von Textdateien einfach ist und man mithilfe von Bibliotheken wie [Matplotlib](#) sehr schnell hochqualitative Diagramme erstellen kann. In dieser Aufgabe möchten wir einige einfache Funktionen in Basic-IO nachprogrammieren, um zu verstehen wie eine solche Bibliothek aufgebaut sein könnte.

**Hinweis:** Sie finden ein Kurzsript zur Verwendung von Matplotlib [hier](#)

Zuerst müssen wir die Daten herunterladen und einlesen.

- Wählen Sie von der [Webseite des Deutschen Wetterdienstes](#) die Option *Tageswerte* und einen Standort aus (z.B. „Frankfurt/Main“) und speichern Sie die Tabelle in einer Textdatei ab. Achten Sie darauf nur die Zeilen mit den Daten zu kopieren und nicht die Spaltennamen, HTML Tags etc. **Anmerkung:** Zum Zeitpunkt der Erstellung des Übungsblattes kann man nicht auf die Tageswerte zugreifen. Daher finden Sie auf LMS eine fertige Tabelle mit den Daten für den Münchener Flughafen.
- Lesen Sie mithilfe von Numpy die Tabelle ein. Verwenden Sie dazu die `np.loadtxt` Methode von Numpy. Sie können nun mit `column = table[:,spaltennummer]` eine bestimmte Spalte extrahieren. **Anmerkung:** in der bereitgestellten Datei haben die relevanten Spalten (siehe unten) Temperatur *TX* und Niederschlagsmenge *RR* folgende Indizes: `tx = table[:,6]` und `rr = table[:,-2]`.

Als erstes möchten wir die Daten betrachten. Als x-Werte kann einfach eine Liste mit aufsteigenden Zahlen verwendet werden (`np.arange`).



Ein einfaches Streudiagramm ohne Achsen

**1a)** Zeichnen Sie die Niederschlagsmenge (Spalte *RR*) als rotes Streudiagramm (engl. Scatterplot) wie in der Abbildung gezeigt mit der `draw_circle` Methode von Basic-IO.



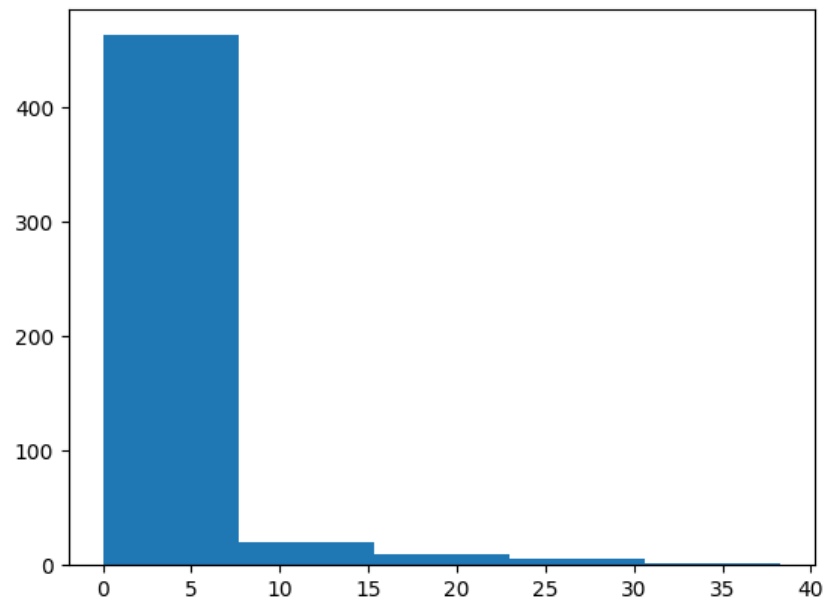
Ein einfaches Liniendiagramm ohne Achsen

**1b)** Zeichnen Sie die Temperatur (Spalte *TX*) als rotes Liniendiagramm (engl. Line Plot) wie in der Abbildung gezeigt mit der `draw_line` Methode von Basic-IO in das gleiche Bild.

**1c)** Fügen Sie Achsen und Achsenbeschriftungen hinzu. Einfache Geraden für die Achsen und Beschriftungen an der richtigen Stelle ohne genaue Kennzeichnungen ("Ticks") sind ausreichend. *Tipp:* Achten Sie auf die Orientierung ihres Diagramms. Der Ursprung des Koordinatensystems von Basic-IO liegt oben links!

**1d)** Importieren Sie Matplotlib mit `import matplotlib.pyplot as plt` und erstellen Sie ein ähnliches Diagramm nun mit der `plot`-Methode von Matplotlib: `plt.plot`. Dies sollte deutlich einfacher sein als alles selbst zu programmieren! Sie können die `linestyle`, `color` und `marker` Argumente anpassen, damit ihr Diagramm die gewünschte Erscheinung bekommt. Beschriften Sie die Achsen ihres Diagramms mit den `plt.xlabel`, `plt.ylabel` Methoden und geben Sie ihrem Histogramm einen Titel mit der `plt.title` Methode.

Wir möchten nun den Niederschlag als Histogramm betrachten.



Ein einfaches Histogramm mit Achsen

**2a)** Ermitteln Sie das Maximum und Minimum des Niederschlags, so erhalten wir den Wertebereich. Teilen Sie den erhaltenen Wertebereich in fünf gleich große Abschnitte (engl. Bins). Ermitteln Sie wieviele Werte in die jeweiligen Abschnitte fallen.

**2b)** Erstellen zu den Niederschlagswerten ein Histogramm(Balkendiagramm) mit Basic-IO. Die Höhe der Balken soll proportional zur Anzahl an Werten im Bereich sein.

**2c)** Fügen Sie Achsen und Achsenbeschriftungen hinzu. Einfache Geraden für die Achsen und Beschriftungen an der richtigen Stelle ohne genaue Kennzeichnungen ("Ticks") sind ausreichend.

**2d)** Erstellen Sie einen Ähnliches Diagramm nun mit der [hist](#)-Methode von Matplotlib. Sie können das `bins` Argument anpassen, damit ihr Diagramm die gewünschte Erscheinung bekommt. Beschriften Sie die Achsen ihres Diagramms mit den `plt.xlabel`, `plt.ylabel` Methoden von Matplotlib.