3. Arbeiten mit Dateien

1 Python Built-In Methode

Python unterscheidet Textdateien und binäre Dateien. Textdateien lassen sich mit der Methode **open()** öffnen.

- **open(filename, mode):** öffnet eine Datei (d.h. erzeugt ein Datei-Objekt), modi: r (read), a (append), w (write), r+ (read write)
- **read():** liest den gesamten Dateieninhalt in einen String ein
- **read(5):** liest die ersten fünf characters des Strings ein
- **readline():** liest eine Zeile bis zum EOL (End of Line) character (z.B. ein newline character: \n (enter)
- **readlines():** liest alle Zeilen der Datei, werden in Liste gespeichert (jede Zeile ein Eintrag)
- **write():** schreibt eine Zeile in die Datei
- **close():** wichtig! um Datei zu schließen und die Ressourcen wieder freizugeben
- **with ... as:** sichere und ßaubere"Methode um eine Datei zu lesen oder zu bearbeiten, schließt die Datei am Ende automatisch

  Beispiel:

```
with open("MyTestFile.txt", "r") as file:
    for line in file:
        line.readline()
```

2 Numpy - Dateien als Array einlesen

- **genfromtxt:**
  - kann fehlende Werte übergehen
  - sehr viele Optionen
  - liest gesamte Datei in den Arbeitsspeicher: kann bei großen Dateien zu Problemen führen
- **loadtxt:**
  - jede Spalte muss gleiche Anzahl an Werten haben
  - hat weniger Optionen
  - liest nicht die gesamte Datei gleichzeitig in den Arbeitsspeicher
- **savetxt:**
  - speichert die Arrays als Textdateien ab
  - hat ebenfalls viele Definitionsmöglichkeiten: delimiter, format, header, ...)

3 Experimentelle Daten

(Die Pythonbefehle in diesem Abschnitt sind als Beispiel für genfromtxt angegeben)

- **Header Footer:** Oft schreiben Geräte sowohl an den Anfang als auch an das Ende der Datei Parameterwerte. Diese müssen übersprungen werden (z.B. mit **skip_header**), um das Datenarray einzulesen
- **comments:** manchmal ist die genaue Zahl der Zeilen als Header unbekannt, dafür wird (hoffentlich) ein gleichbleibendes Kommentarzeichen verwendet, was mit **comments** deklariert werden kann
- **delimiter:** der Character, der die einzelnen Einträge trennt, kann über **delimiter** deklariert werden. Oft wird es aber auch richtig erkannt. Der angegebene Delimiter kann ein Zeichen sein (z.B. \t), oder die Breite des Eintrags, oder eine Sequence
- **dtype:** Deklarierung des zu erwartenden Datentyps

4 Darstellung der Daten: Matplotlib.pyplot

- Standardbibliothek für viele Plots, angelehnt an MATLAB
- Jedes Objekt in einem Plot ist ein Pythonobjekt und kann als solches manipuliert werden (Achsenbeschriftung, Ticks, Layout,...)
- umfassende Objekt des gesamten Plots: **figure**
- Simplester Plotaufruf:

```
import matplotlib.pyplot as plt

plt.plot(x, y)
plt.savefig("MyPlot.png")
plt.show()
```

- Anatomie einer Figure: Link
- Hierarchie der Matplotlib-Umgebung: