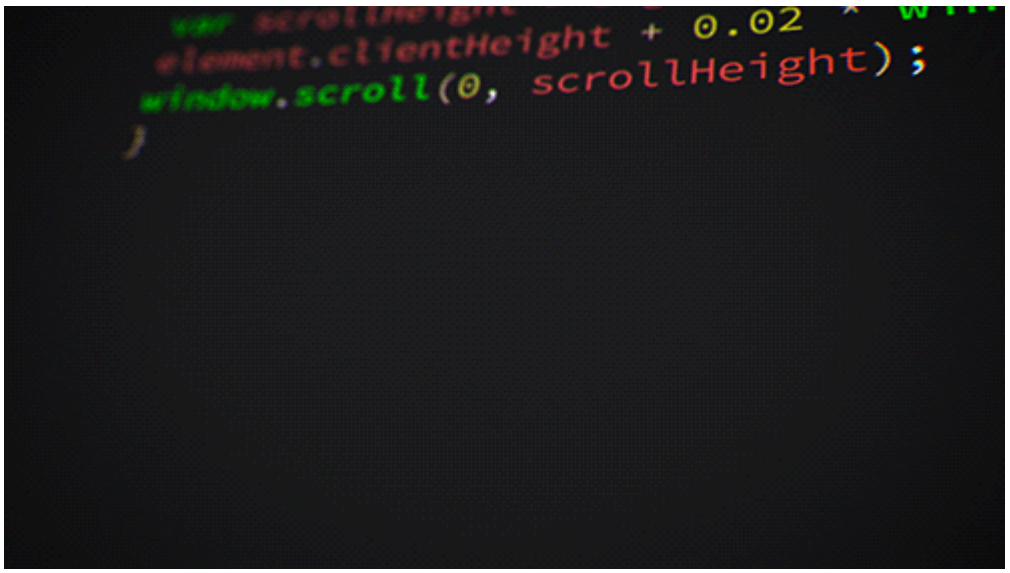
Datenanalyse mit Python

Parameter	Kursinformationen
Veranstaltung:	`Prozedurale Programmierung / Einführung in die Informatik / Erhebung, Analyse und Visualisierung digitaler Daten`
Semester	`Wintersemester 2025/26`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	`Datenanalyse mit dem Python Paket Pandas`
Link auf Repository:	https://github.com/TUBAF-IfI-LiaScript/VL_EAVD/blob/master/10_DatenAnalyse.md
Autoren	Sebastian Zug & André Dietrich & Galina Rudolf



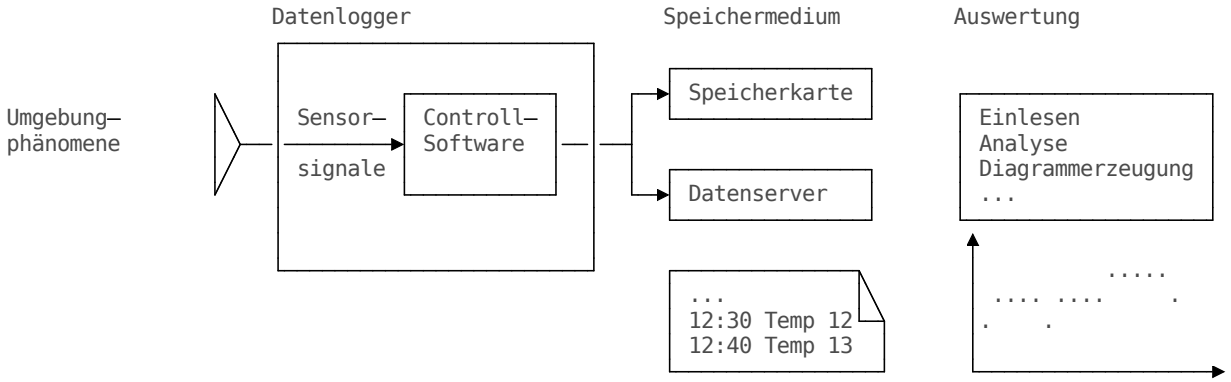
Fragen an die heutige Veranstaltung ...

- Welche Datenformate sind für den Daten Austausch zwischen Mikrocontroller und Python Script üblich?
- Wie unterstützt das `pandas` Paket die wissenschaftliche Analyse von Datensätzen?
- Wie ändert sich der Analyseprozess verglichen mit der Verwendung einer Tabellenkalkulation?

Motivation

Aufgabe: Dokumentieren Sie die zeitliche Verteilung des Erscheinens von Vögeln an einer Futterstelle. Zu welcher Tages / Nachtzeit ist die Aktivität am größten?

Vogelhaus (Fotograph Sagar Kumar Singh Pexels)



Und der Code für den Datenlogger? Wir werten den Beschleunigungssensor unseres Controllerboards aus.

```
#include "Sensor.h"
#include "RGB_LED.h"

DevI2C i2c(D14,D15);
LSM6DSLSensor sensor(i2c,D4,D5);
int xAxesData[3];
```

```
void setup() {
    Serial.begin(115200);          //Baudrate der Seriellen Schnittstelle
    sensor.init(NULL);             //Start des Sensors
    sensor.enableAccelerator();    //Aktivierung des Beschleunigungssensors
}

void loop() {
    sensor.getXAxes(xAxesData);  //Lesen der Daten
    Serial.printf("; %d; %d; %d\n",xAxesData[0],xAxesData[1],xAxesData[2]);
                                 //Ausgabe der Daten via Serielle
                                 //Schnittstelle
    delay(10);
}
```

> Aufgabe: Bewerten Sie die Implementierung! Welche Nachteile sehen Sie?

Für die Konfiguration des Zeitstempels im Visual Studio Code wurde der Parameter *Arduino: Change Timestamp Format* `%T.%L` strftime Format gesetzt.

Die Daten liegen als sogenannten *Comma-separated values* in einer csv Datei vor. Sie sind eine bequeme Möglichkeit, Daten aus Tabellenkalkulationen und Datenbanken zu exportieren und sie in andere Programme zu importieren oder zu verwenden. CSV-Dateien lassen sich sehr einfach programmatisch bearbeiten. Jede Sprache, die die Eingabe von Textdateien und die Manipulation von Zeichenketten unterstützt (wie Python), kann direkt mit CSV-Dateien arbeiten. Nachteilig ist, dass alle Inhalte als Text angelegt werden und damit verschwenderisch mit dem Speicher umgehen.

Als Trenner wurde hier das `;` verwendet.

```
09:28:52.419; -7; -8; 1016
09:28:52.430; -9; -8; 1017
09:28:52.441; -9; -8; 1017
09:28:52.452; -9; -8; 1017
09:28:52.463; -16; -2; 1006
09:28:52.474; -69; -160; 1057
09:28:52.485; 58; 136; 984
09:28:52.496; -10; -10; 1019
09:28:52.507; -11; -6; 1012
09:28:52.518; -5; 0; 1016
09:28:52.528; -9; -15; 1013
09:28:52.539; -9; -8; 1018
09:28:52.551; -8; -9; 1016
09:28:52.562; -8; -9; 1019
```

- Wie groß ist das normale Rauschen der Messwerte?
- Wann wurde die größte Änderung der Beschleunigung gemessen?
- Stellen Sie die Verlauf in einem Diagramm dar, benennen Sie die Achsen, erzeugen Sie ein Gitter.

MEMO! Arbeiten Sie bei der Analyse immer auf Kopien der eigentlichen Daten. Im Fall einer "Kompromitierung" durch eine einfache Schreiboperation haben Sie immer noch den Originaldatensatz zur Verfügung.

Lösungsansatz 1: Office Tabellenkalkulation

data.csv

```
timestamp;X;Y;Z
09:28:52.419; -7; -8; 1016
09:28:52.430; -9; -8; 1017
09:28:52.441; -9; -8; 1017
09:28:52.452; -9; -8; 1017
```

Nutzen Sie die Importfunktion für csv-Dateien

Lösungsansatz 2: Python nativ

Python kann die Textdateien umittelbar einlesen

1. Öffnen der Datei für das Lesen
2. Zeilenweises einlesen der Daten
   - Erfassen der Spaltennamen aus der ersten Zeile
   - Zerlegen anhand des `delimiter` (hier `;`)
   - Ablegen in einer vorbereiteten Datenstruktur
3. Schließen der Datei
4. Analyse der Daten

Diese Schrittfolge können wir mit dem Standardpaket csv etwas vereinfachen.

data.csv

```
timestamp;X;Y;Z
09:28:52.419; -7; -8; 1016
09:28:52.430; -9; -8; 1017
09:28:52.441; -9; -8; 1017
09:28:52.452; -9; -8; 1017
```

readCSV.py

```python
import csv

# Einlesen der Daten
with open('data.csv', mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file, delimiter=';',)
    list_of_dict = list(csv_reader)

# "Analyse" und Ausgabe
print(f"{len(list_of_dict)} Datensätze gelesen!")
for row in list_of_dict:
  print(row)

csv_file.close()
```

```
Waking up execution server ...
This may take up to 30 seconds ...
Please be patient ...
.......................
```

Aufgabe: Bestimmen Sie die vorkommenden Maxima pro Spalte oder berechnen Sie die Differenz zwischen zwei benachbarten Werten einer Beschleunigungsachse.

Lösungsansatz 3: Python mit Pandas

pandas ist eine für die Programmiersprache Python geschriebene Softwarebibliothek zur Datenmanipulation und -analyse, die insbesondere Datenstrukturen und Operationen zur Manipulation von numerischen Tabellen und Zeitreihen bietet. Es handelt sich um freie Software.

Der Name leitet sich von dem Begriff "*panel data*" ab, einem Begriff aus der Ökonometrie für Datensätze, die Beobachtungen über mehrere Zeiträume für dieselben Personen enthalten.

Der Code zum Paket kann unter Link eingesehen und bearbeitet werden.

Achtung: Mit der Verwendung von pandas ändert sich unser Blick auf den Code. Bislang haben wir Prozedural oder Objektorientiert programmiert. Jetzt ändert sich unser Blick - wir denken in Datenstrukturen und wenden Methoden darauf an.

Pandas Grundlagen

Pandas kennt zwei grundsätzliche Datentypen Series und DataFrame

	Pandas Series	**Pandas DataFrame**
Format	Eindimensional	Zweidimensional
Datentypen	Homogen - Reihenelemente müssen vom gleichen Datentyp sein.	Heterogen - DataFrame-Elemente können unterschiedliche Datentypen haben.
Zahl der Einträge	Größenunveränderlich - Einmal erstellt, kann die Größe nicht geändert werden.	Größenveränderlich - Elemente können in einem bestehenden DataFrame gelöscht oder hinzugefügt werden.

Wir betrachten zunächst die grundsätzliche Arbeitsweise für Series Daten.

PandasSeries.py

```
1  import pandas as pd
2  import numpy as np
3
4  #Zufallszahlen mit dem Paket "numpy"
5  s_1 = pd.Series(np.random.randn(5))
6  print(s_1)
7
8  #Zufallszahlen und individuelle Indizes
9  s_2 = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])
10 print(s_2)
11
12 # Für unseren Datensatz und die Z Beschleunigungsdaten
13 data = {"09:28:52.419": 1016, "09:28:52.430": 1017, "09:28:52.441": 1
14 s_3 = pd.Series(data)
15 print(s_3)
```

```
.........................
```

Achtung: Im letztgenannten Beispiel `s_3` werden die Indizes nicht als Datum interpretiert sonder als Text. Realistisch wäre hier noch eine Transformation notwendig!

PandasDataFrame.py

```python
import pandas as pd
import numpy as np

#Multidimensionales DataFrame mit identischen Datentypen
df_1 = pd.DataFrame(np.random.randn(6, 4))
print(df_1)
print()

#Variables Set von Daten unterschiedlicher Typen
df_2 = pd.DataFrame(
    {
        "A": True,
        "B": pd.date_range("20230101", periods=4),
        "C": pd.Series(np.random.randn(4)),
        "D": np.random.randint(16, size=4),
        "E": pd.Categorical(["A", "A", "B", "C"]),
        "F": "foo",
    }
)
print(df_2)
```

```
.....................
```

Aufgabe: Evaluieren Sie mittel `print(df_2.dtypes)` die realsierten Datentypen für `df_2`. Worüber "stolpern" Sie?

Arbeit mit Dataframes

Welche Aufgaben lassen sich nun mit Hilfe von Pandas über den Daten realisieren?

Indizierung

data.csv

```
timestamp;X;Y;Z
09:28:52.353; -8; -9; 1016
09:28:52.364; -9; -8; 1017
09:28:52.375; -9; -8; 1017
09:28:52.386; -8; -8; 1016
09:28:52.397; -9; -8; 1017
09:28:52.408; -9; -8; 1018
09:28:52.419; -9; -8; 1016
09:28:52.430; -9; -8; 1017
09:28:52.441; -9; -8; 1017
09:28:52.452; -9; -8; 1017
```

index.py

```
import pandas as pd

df = pd.read_csv('data.csv', header = 0, sep=";")
print(df)
```

```
.........................
```

Filtern

data.csv

```
timestamp;X;Y;Z
09:28:52.353; -8; -9; 1016
09:28:52.364; -9; -8; 1017
09:28:52.375; -9; -8; 1017
09:28:52.386; -8; -8; 1016
09:28:52.397; -9; -8; 1017
09:28:52.408; -9; -8; 1018
09:28:52.419; -9; -8; 1016
09:28:52.430; -9; -8; 1017
09:28:52.441; -9; -8; 1017
09:28:52.452; -9; -8; 1017
```

filter.py

```
import pandas as pd

df = pd.read_csv('data.csv', header = 0, sep=";")
print(df)
```

```
.......................
```

Statistische Beschreibung

data.csv

```
timestamp;X;Y;Z
09:28:52.353; -8; -9; 1016
09:28:52.364; -9; -8; 1017
09:28:52.375; -9; -8; 1017
09:28:52.386; -8; -8; 1016
09:28:52.397; -9; -8; 1017
09:28:52.408; -9; -8; 1018
09:28:52.419; -9; -8; 1016
09:28:52.430; -9; -8; 1017
09:28:52.441; -9; -8; 1017
09:28:52.452; -9; -8; 1017
```

describe.py

```
import pandas as pd

df = pd.read_csv('data.csv', header = 0, sep=";")
print(df.describe())
```

```
.......................
```

Nutzung

data.csv

```
timestamp;X;Y;Z
09:28:52.353; -8; -9; 1016
09:28:52.364; -9; -8; 1017
09:28:52.375; -9; -8; 1017
09:28:52.386; -8; -8; 1016
09:28:52.397; -9; -8; 1017
09:28:52.408; -9; -8; 1018
09:28:52.419; -9; -8; 1016
09:28:52.430; -9; -8; 1017
09:28:52.441; -9; -8; 1017
09:28:52.452; -9; -8; 1017
```

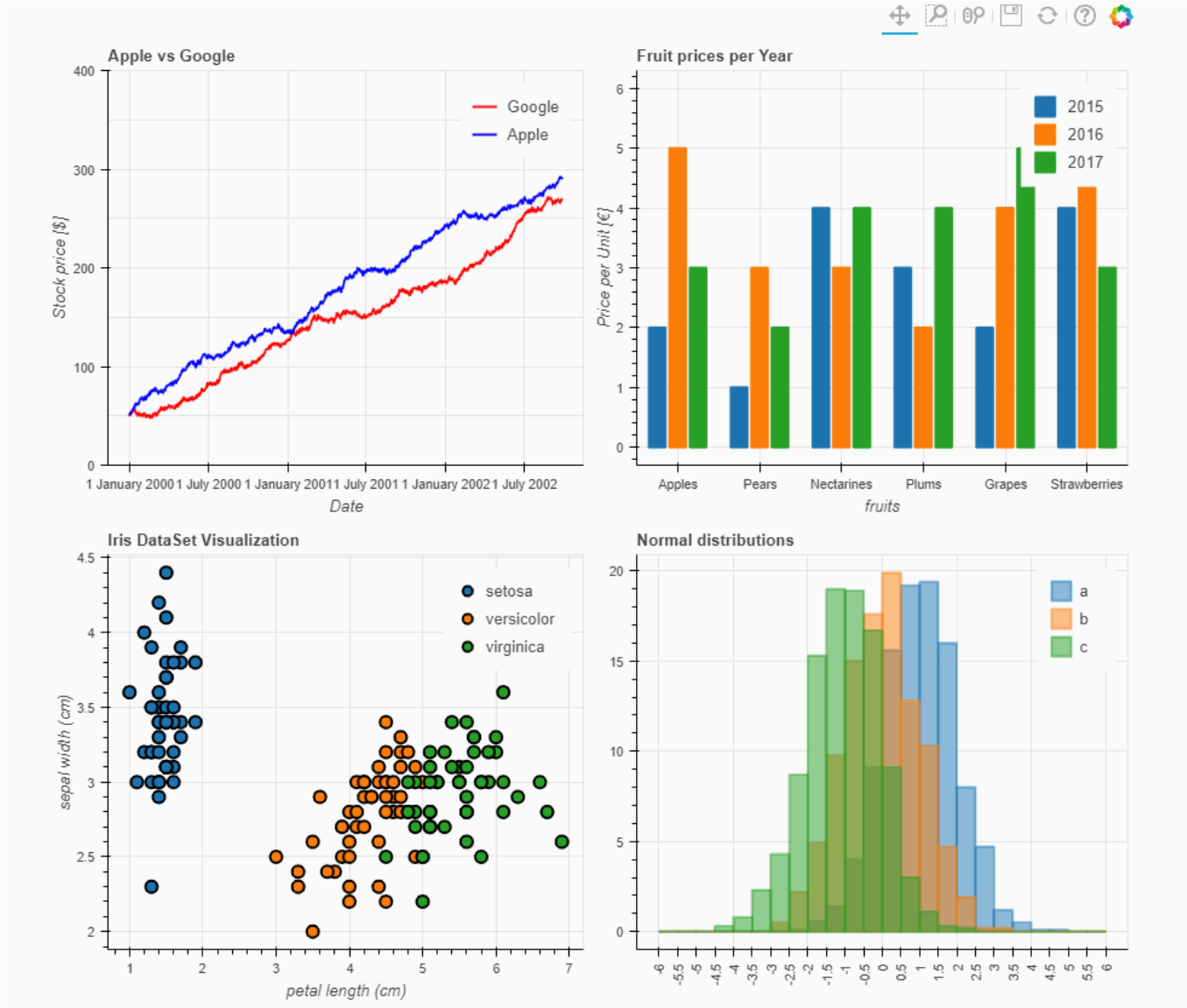
describe.py

```
1  import pandas as pd
2
3  df = pd.read_csv('data.csv', header = 0, sep=";")
4  print(df)
```

```
.........................
```

Visualisierung mit pandas

Pandas ist unmittelbar mit der schon bekannten Bibliothek matplotlib verknüpft. Damit können wir unsere bereits bekannten Methoden nahtlos nutzen.



Beispiele der Visualisierung von Pandas 'PatrikHlobil' [Link]

data.csv

```
timestamp;X;Y;Z
09:28:52.353; -8; -9; 1016
09:28:52.364; -9; -8; 1017
09:28:52.375; -9; -8; 1017
09:28:52.386; -8; -8; 1016
09:28:52.397; -9; -8; 1017
09:28:52.408; -9; -8; 1018
09:28:52.419; -9; -8; 1016
09:28:52.430; -9; -8; 1017
09:28:52.441; -9; -8; 1017
09:28:52.452; -9; -8; 1017
09:28:52.463; -16; -2; 1006
09:28:52.474; -69; -160; 1057
09:28:52.485; 58; 136; 984
09:28:52.496; -10; -10; 1019
09:28:52.507; -11; -6; 1012
09:28:52.518; -5; 0; 1016
09:28:52.528; -9; -15; 1013
09:28:52.539; -9; -8; 1018
09:28:52.551; -8; -9; 1016
09:28:52.562; -8; -9; 1019
09:28:52.572; -8; -8; 1015
09:28:52.583; -8; -8; 1015
09:28:52.595; -9; -7; 1017
09:28:52.606; -9; -8; 1016
09:28:52.617; -8; -9; 1016
09:28:52.628; -7; -9; 1018
```

readCSV.py

```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv', header = 0, sep=";")
df.plot()
plt.savefig('foo.png')
```

```
.........................
```

Anpassung	API	
Linientyp der Datendarstellung	pyplot.plot	`plt.plot(a, b, 'ro:')`
Achsenlabel hinzufügen	pyplot.xlabel	`plt.xlabel('my data', fontsize=14, color='red')`
Titel einfügen	pyplot.title	`plt.title(r'$\sigma_i=15$')`
Gitter einfügen	pyplot.grid	`plt.grid()`
Legende	pyplot.legend	`plt.plot(a, b, 'ro:', label="Data")`
		`plt.legend()`
Speichern	pyplot.savefig	`plt.savefig('foo.png')`

Merke: Mit dem zusätzlichen Parameter `styple='o:'` können Sie die Konfiguration der Darstellung anpassen.

Aufgabe 1: Weisen Sie grafisch nach, dass es einen starken Zusammenhang zwischen den 3 Beschleunigungsdaten gibt!

Aufgabe 2: Geben Sie die Daten einer Achse in einem Histogramm aus! Schreiben Sie als Text den maximalen und den Minimalen Wert in die Mitte des Diagrams.

data.csv

```
timestamp;X;Y;Z
09:28:52.353; -8; -9; 1016
09:28:52.364; -9; -8; 1017
09:28:52.375; -9; -8; 1017
09:28:52.386; -8; -8; 1016
09:28:52.397; -9; -8; 1017
09:28:52.408; -9; -8; 1018
09:28:52.419; -9; -8; 1016
09:28:52.430; -9; -8; 1017
09:28:52.441; -9; -8; 1017
09:28:52.452; -9; -8; 1017
09:28:52.463; -16; -2; 1006
09:28:52.474; -69; -160; 1057
09:28:52.485; 58; 136; 984
09:28:52.496; -10; -10; 1019
09:28:52.507; -11; -6; 1012
09:28:52.518; -5; 0; 1016
09:28:52.528; -9; -15; 1013
09:28:52.539; -9; -8; 1018
09:28:52.551; -8; -9; 1016
09:28:52.562; -8; -9; 1019
09:28:52.572; -8; -8; 1015
09:28:52.583; -8; -8; 1015
09:28:52.595; -9; -7; 1017
09:28:52.606; -9; -8; 1016
09:28:52.617; -8; -9; 1016
09:28:52.628; -7; -9; 1018
```

ScatterPlot.py

```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv', header = 0, sep=";")
# Hier sind sie gefragt ...
plt.savefig('foo.png')
```

```
.......................
```

Noch immer von Excel überzeugt?

- **Skalierbarkeit** - Pandas ist nur durch die Hardware begrenzt und kann größere Datenmengen verarbeiten. Excel ist aktuell auf 1.048.576 Zeilen und 16.384 Spalten beschränkt.
- **Geschwindigkeit** - Pandas arbeitet viel schneller als eine Tabellenkaklulation, was sich besonders bei der Arbeit mit größeren Datenmengen bemerkbar macht.
- **Automatisierung** - Viele der Aufgaben, die mit Pandas durchgeführt werden können, sind extrem einfach zu automatisieren, wodurch die Anzahl der langweiligen und sich wiederholenden Aufgaben, die täglich durchgeführt werden müssen, reduziert wird.
- **Interpretierbarkeit** - Eine Codesequenz aus Pandas ist übersichtlich und einfach zu interpretieren, da Tabellenkalkulationen Berechnungen pro Zelle ausführen, sind Fehler schwieriger zu identifizieren und zu beheben.
- **Erweiterte Funktionen** - Die Durchführung erweiterter statistischer Analysen und die Erstellung komplexer Visualisierungen ist sehr einfach.

pandas	Tabellenkalkulation
`DataFrame`	*worksheet*
`Series`	*column*
`Index`	*row headings*
`row`	*row*
`NaN`	empty cell

```
# Einlesen einer Excel Datei in Pandas
df = pd.read_excel("./myExcelFile.xlsx", index_col=0)

# Schreiben einer Excel Datei aus Pandas
df.to_excel("./myExcelFile.xlsx")
```

Beispiel der Woche

Der Deutsche Wetterdienst bietet auf seinen Webseiten eine vielzahl von historischen Datensätzen. Wir wollen unsere Pandas Kenntnisse nutzen, um uns darin zu orientieren und dann "Licht in den Nebel bringen".

Die historischen Aufzeichnungen zu verschiedenen Stationen in Deutschland finden sich in der Datenbank.

Aufgabe 1: Finden Sie die Stationsnummern von sächsischen Stationen in der Übersicht der Wetterstationen.

Den Originaldatensatz des deutschen Wetterdienstes können wir nicht verwenden - dieser ist als csv nicht ohne größeren Aufwand zu lesen. Daher wurde diese Datei aus didaktischen Gründen angepasst und liegt im Repository unter der angegebenen URL bereit.

findweatherstations.py

```
1  import pandas as pd
2
3  url="https://raw.githubusercontent.com/" + \
4      "TUBAF-IfI-LiaScript/VL_ProzeduraleProgrammierung/" + \
5      "master/examples/11_DatenAnalyse/" + \
6      "Wetterdaten/wetter_tageswerte_Beschreibung_Stationen.txt"
7
8  df=pd.read_csv(url, sep=';', header = 0)
9  #df=pd.read_csv("wetter_tageswerte_Beschreibung_Stationen.txt", sep='
     header = 0)
10 df['Bundesland'] = df['Bundesland'].str.strip()
11 df['Stationsname'] = df['Stationsname'].str.strip()
12 df_sachsen = df[df.Bundesland  == "Sachsen"]
13 print(df_sachsen)
```

```
.........................
```

Leider reicht der Freiberger Datensatz nur über wenige Jahre. Wir entscheiden uns für die weitere Untersuchung für die Daten aus Görlitz.

Aufgabe 2: Laden Sie den Görlitzer Datensatz in einen Dataframe und zählen Sie die Nebeltage. Visualisieren Sie das Ergebnis.

Der avisierte Datensatz für die Station "1684" heruntergeladen. Die Datei `wetter_tageswerte_01684_18800101_20181231_hist.zip` liefert als gepackter Ordner mehrere Datensets. Neben der eigentlichen Datendatei werden auch Stationskerndaten und Erhebungstechnik dokumentiert.

extractFogObservations.py

```python
import pandas as pd
import matplotlib.pyplot as plt

url="https://raw.githubusercontent.com/" + \
    "TUBAF-IfI-LiaScript/VL_ProzeduraleProgrammierung/" + \
    "master/examples/11_DatenAnalyse/" + \
    "Wetterdaten/" + \
    "wetter_tageswerte_01684_18800101_20181231_hist/" + \
    "produkt_wetter_tag_18800101_20181231_01684.txt"

df=pd.read_csv(url, sep=';', header = 0)

df["JAHR"]=df["MESS_DATUM"]/10000
df["JAHR"] = df["JAHR"].astype('int')

df_fog = df[df.NEBEL!=-999]

print(f"{df_fog.NEBEL.count()} Tage im Datensatz, {df_fog.NEBEL.sum()}
  Nebel.")

ax = plt.axes()
df_fog.groupby("JAHR").NEBEL.sum().plot.bar(ax=ax)
ax.xaxis.set_major_locator(plt.MaxNLocator(20))
plt.ylabel("Häufigkeit von Nebelbeobachtungen")
plt.title("Nebeltage über Görlitz")
plt.grid()

#plt.show()
plt.savefig('foo.png') # notwendig für die Ausgabe in LiaScript
```

```
.........................
```

Aufgabe: Evaluieren Sie, welche Parameter sich in den vergangenen Jahren signifikant verändert haben.

Quiz

CSV-Dateien

Wofür steht *CSV*?

- *Common System Variables*
- *Colorful Systematic Visualization*
- *Comma-separated values*
- *Critical Signal Version*

Python nativ

Wie lautet die Ausgabe des folgenden Programms, das die Daten aus der Datei data.csv einliest?

data.csv

```
timestamp;X;Y;Z
09:28:52.419;-7;-8;1016
09:28:52.430;-9;-8;1017
09:28:52.441;-9;-8;1017
09:28:52.452;-9;-8;1017
```

```
import csv

with open('data.csv', mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file, delimiter=';',)
    list_of_dict = list(csv_reader)

for row in list_of_dict:
  print(row['X'], end=",")

csv_file.close()
```

Pandas Grundlagen

Ordnen Sie *Pandas Series* und *Pandas Dataframes* die richtigen Eigenschaften zu.

Pandas Series	*Pandas Dataframe*	
◯	◯	Eindimensional
◯	◯	Zweidimensional
◯	◯	Heterogene Datentypen
◯	◯	Homogene Datentypen
◯	◯	Größenunveränderlich
◯	◯	Größenveränderlich

Arbeit mit Dataframes

Wie lautet die Ausgabe dieses Programms?

```
import pandas as pd

a = [5, 7, 2, 7, 6]

s_1 = pd.Series(a, index=["a", "b", "c", "d", "e"])
print(s_1["c"])
```

Wie lautet die Ausgabe dieses Programms?

```
import pandas as pd
```

```
d = {
    'A': [1,4,2,5],
    'B': [2,5,3,6],
    'C': [3,6,4,7],
    'D': [4,7,5,8]
}

s_1 = pd.Series(d)
print(s_1['C'][2])
```

Wie lautet die Ausgabe dieses Programms?

data.csv

```
title,FSK
Toy Story,0
Jumanji,12
Grumpier Old Men,6
Waiting to Exhale,12
Father of the Bride Part II,0
Heat,16
Sabrina,6
Tom and Huck,6
Sudden Death,16
GoldenEye,16
The Amerian President,6
Dracula: Dead and Loving It,12
Balto,0
Nixon,12
Cutthroat Island,12
```

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df['title'][9])
```