HSLU Lucerne University of Applied Sciences and Arts

Python for Data Science: SW10

Must have Data Science Packages

Information Technology

April 24, 2025

FH Zentralschweiz

Content

- Data Manipulation
  - Numpy
  - Pandas
- Data Visualization
  - Matplotlib
  - Seaborn
- Jupyter Notebook
  - Inline Image
  - Export to PDF, HTML
- Exercise

HSLU Lucerne University of Applied Sciences and Arts

Data Manipulation

Information Technology

April 24, 2025

Data Manipulation: Numpy

Data scientists typically deal with different kind of data, including categorical values, names (strings) or numerical values. For numerical values, NumPy is the standarad way to go.

"NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a **multidimensional array object**, various derived objects (such as masked arrays and matrices), and an assortment of **routines for fast operations on arrays**, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more."

Documentation (API reference): https://numpy.org/doc/stable/reference/index.html

Data Manipulation: Numpy

Multidimensional array object (`ndarray`) are sequences, matrices or multidimensional matrices of numerical values.

171	45	172	107	164	88	110	9	168	125	87	117	14	115	183	54	70

Important differences between **NumPy arrays** and the standard Python **sequences** are:

- Fixed size at creation. Changing size of `ndarray` creates new array.
- All elements are required to be of the same data type.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, it executes operations more efficiently and with less code.

171	110
45	9
172	168
107	125
164	87
88	117

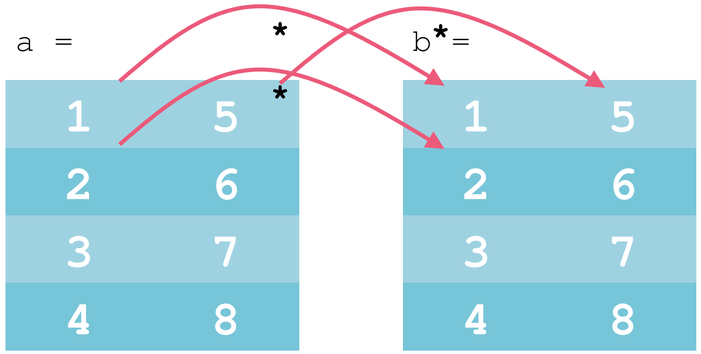
Data Manipulation: Numpy

An example for more efficient and convenient way working with ndarray compared to built-in sequences is given in element-by-element operation.

Assume two lists of same size have to be muliplied element-wise:

- Built-in Python

```
>>> a = b = c = [[1,2,3,4],[5,6,7,8]]
>>> for i in range(len(a)):
...     for j in range(len(a[i])):
...         c[i][j] = a[i][j] * b[i][j]
...
>>> c
[[1, 4, 9, 16], [25, 36, 49, 64]]
```



- Numpy

```
>>> import numpy
>>> a = b = numpy.array([[1,2,3,4],[5,6,7,8]])
>>> c = a*b
>>> c
array([[ 1,  4,  9, 16],
       [25, 36, 49, 64]])
>>>
```

Data Manipulation: Pandas

For data handling, data scientists often work with **data frames**. These allow data scientists to bring data in a structured order that facilitates data analysis and manipulation. In Python, data frames are provided by the **pandas** library.

pandas is a fast, powerful, flexible and easy to use open source **data analysis** and **manipulation tool**,

built on top of the Python programming language.

Documentation (API reference): https://pandas.pydata.org/docs/reference/index.html#api

CheatSheet: https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

Data Manipulation: Pandas

The Pandas library is a powerful and widely-used tool in Python for **data manipulation** and **analysis**. It provides a variety of features and functionalities, including:

- A fast and efficient DataFrame object for data manipulation with integrated indexing.
- **Reading and writing** data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format.
- Intelligent data alignment and integrated handling of **missing data**.
- Flexible reshaping and pivoting of data sets.
- Intelligent **label-based** slicing, fancy indexing, and subsetting of large data sets.
- Columns can be inserted and deleted from data structures for **size mutability**.
- Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets.
- High performance **merging** and **joining** of data sets.
- **Time series-functionality**: date range generation, frequency conversion, moving window stats, date shifting and lagging.

Data Manipulation: Pandas

Data frame: tabular data structure commonly used in data analysis and statistical computing.

- Columns represent variables (features).
- Rows represent observations (records).
- Indexing by labels or numerical indices: [row, col]
- Data frames can combine heterogenous data types.
- One data type per column.
  -> heterogenous data in one column result
  in “object” data type.

[,col]

[row,]

	V1	V2	V3
0	11	Lucerne	LU
1	321	Bern	BE
2	53	Valais	VS
3	4	Zug	ZG
4	67	Geneva	GE

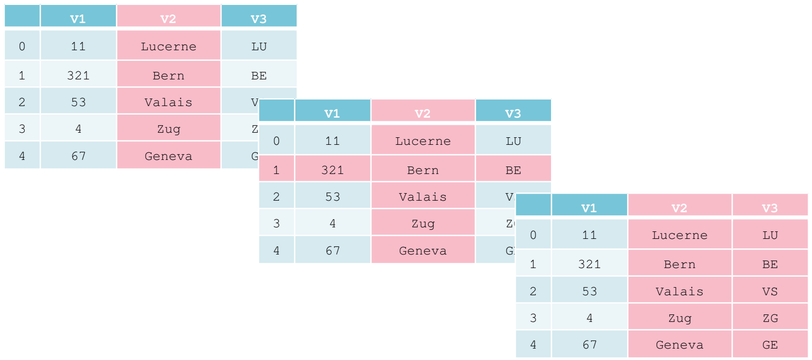
- A data frame can be created from a dictionary:

```
df = pandas.DataFrame({“V1”:[11,321,53,4,67], “V2”:[“Lucerne”, “Bern”, ...], “V3”:[...]})
```

Series: one dimensional data (index and one column of a data frame).

Data Manipulation: Pandas

Dealing with data frames often implies to apply functions to all (or multiple) observations or variables. With pandas data frames, this can be achieved by the functions:



- **map**(): is used for element-wise operations on **Data Frames** and **Series**.
- **apply**(): is used to apply a function along a particular **axis** of a DataFrame.

In a data frame, basic mathematical operations can directly be applied on variables.

Instead using for- or while-loops for iterating over a sequence, with data frames these functions do the same job.

```
>>> for i in df["V2"]:
...     i.upper()
...
'LUCERNE'
'BERN'
'VALAIS'
'ZUG'
'GENEVA'
```

```
>>> df["V2"].map(str.upper)
0    LUCERNE
1       BERN
2     VALAIS
3        ZUG
4     GENEVA
Name: V2, dtype: object
```

HSLU Lucerne University of Applied Sciences and Arts

Data Visualization

Information Technology

April 24, 2025

Data Visualization

Python provides several packages for data visualization, depending on the objectives. Two famous packages for data visualization are:

- Matplotlib: a low-level library that provides a lot of flexibility and control over the creation of plots.
  - Offers fully customizable figures.

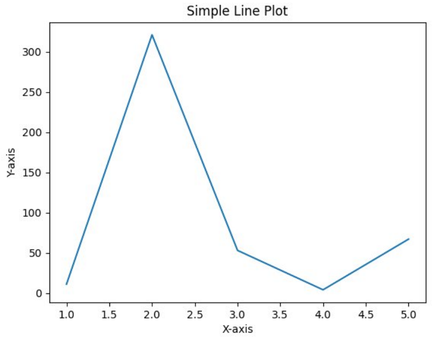
- Seaborn: built on top of Matplotlib, Seaborn is a high-level library specifically designed for statistical data visualization.
  - offers a variety of built-in themes, color palettes and styles.
  - designed to work seamlessly with Pandas DataFrames.

Data Visualization: Matplotlib

Matplotlib has three fundamental components used for creating and managing plots: **Pyplot**, **Figure object** and **Axes object**.

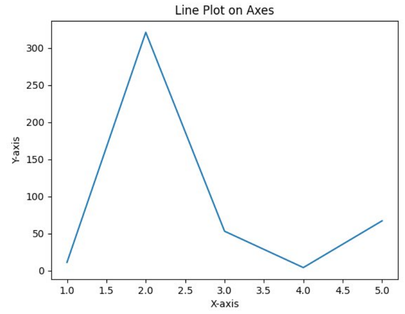
- **Pyplot**: a collection of functions to create figures.

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,4,5], df["V1"])
plt.title("Simple Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```




```
# Create a figure object
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)  # 1 row, 1 column, 1st subplot

# Plot data on the axes
ax.plot([1,2,3,4,5], df["V1"] )
ax.set_title("Line Plot on Axes")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
```

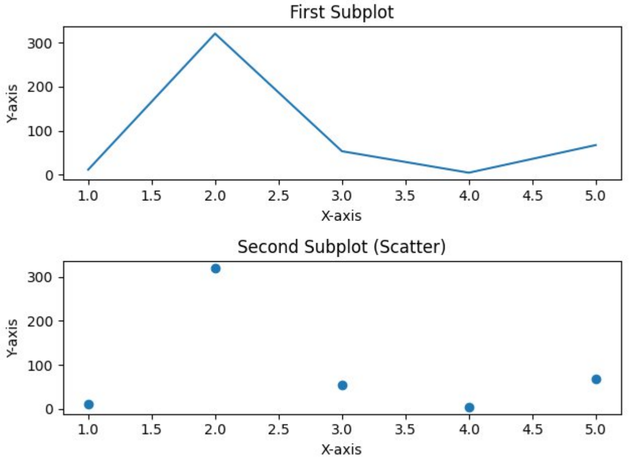
- **Figure object**: represents the entire figure in which you can plot **one** or **more axes**. It is like a canvas on which your plots are drawn.



Data Visualization: Matplotlib

- **Axes object**: represents a single or a set of plots within a figure. This is the area where data is plotted. The axes objects is able to create multiple axes in a single figure, which allows you to create complex layouts of plots.

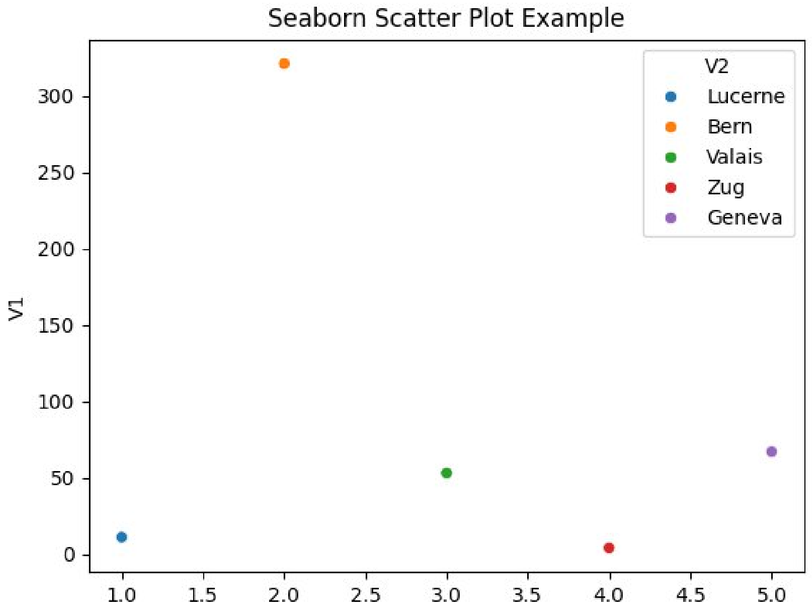
```
# Create a figure with multiple subplots
fig, axs = plt.subplots(2, 1)  # 2 rows, 1 column

# First subplot
axs[0].plot([1,2,3,4,5], df["V1"])
axs[0].set_title("First Subplot")
axs[0].set_xlabel("X-axis")
axs[0].set_ylabel("Y-axis")

# Second subplot
axs[1].scatter([1,2,3,4,5], df["V1"])
axs[1].set_title("Second Subplot (Scatter)")
axs[1].set_xlabel("X-axis")
axs[1].set_ylabel("Y-axis")

plt.tight_layout()  # Adjusts layout to prevent overlap
plt.show()
```



Data Visualization: Seaborn

Since seaborn builds on top of matplotlib, it is **mandatory** to import **pyplot** from matplotlib in order to add titles or show the plot directly.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a scatter plot
sns.scatterplot(data=df, x=[1,2,3,4,5], y='V1', hue='V2')

# Show the plot
plt.title('Seaborn Scatter Plot Example')
plt.show()
```



HSLU Lucerne University of Applied Sciences and Arts

Jupyter Notebook

Information Technology

April 24, 2025

Jupyter Notebook

Jupyter Notebook is an open-source **web application** that allows you to create and share documents that contain **code**, **equations**, **visualizations**, and narrative **text**. It is widely used for data analysis, scientific research, machine learning, and educational purposes.

The main features of Jupyter Notebook are:

- **Interactive Code Execution**: write and execute code in a cell-based format, allowing for iterative development and testing.
- **Rich Text Support**: include Markdown for formatting text, including headings, lists, links, and images.
  (Markdown: https://www.markdownguide.org/)
- **Visualizations**: create and display plots and figures **inline**, making it easy to visualize data alongside your code.
- **Report**: export the notebook as **HTML** or **PDF**, allows for having a report always ready.
  in terminal: `jupyter nbconvert —to html —no-input path/to/notebook.ipynb`
  -> Jupyter and pandoc packages must be installed

Exercise

Exercise: Jupyter Notebook

Often it is more straight forward to analyze data in a jupyter notebook than in an over-engineered python application. Hence, for training purposes, do the following analysis on air quality data given in airquality.csv in a jupyter notebook:

1. Create a new jupyter notebook in PyCharm.
2. Import numpy and pandas and load the airquality.csv file into a data.frame.
3. Calculate some statistics of the temperature values. Report them as text in the jupyter notebook.
4. Find outlier (one) and correct it by averaging the values before and after the outlier. Make a note to the report.
5. Plot the temperature series with matplotlib or seaborn and add it to the report.
6. Create a temperature converter function from °F to °C (formula: °F = °C × (9/5) + 32).
7. Convert all temperature values from °F to °C using **map**().
8. Plot both temperature graphs in **one** figure and add it to the report.
9. Declare a class “AirQualityReport” comprising object variables that store the mean values of columns where reasonable.
10. Add a object method to the class that calculate the max value of all object variables. The function should return a dictionary containing the original column name of the variable with the max value and the maximum value itself.
11. Export the report to an HTML and/or a PDF document.
12. Explore Pandas, Numpy and matplotlib functionalities in more detail.

HSLU Lucerne University of Applied Sciences and Arts

School of Computer Science and Information Technology
Research
Ramón Christen
Research Associate Doctoral Student

Phone direct +41 41 757 68 96
ramon.christen@hslu.ch

HSLU T&A, Competence Center Thermal Energy Storage
Research
Andreas Melillo
Lecturer

Phone direct +41 41 349 35 91
andreas.melillo@hslu.ch