Plotting

Plotting the right information in an easily readable manner is an art. Nonetheless, there are some elements that MUST be present for the plot to be meaningful. Make sure that your plots contain the following information (these are the essentials of a good plot):

- names for axes
- units
- legends (especially if there is more than one data series)
- title

Additionally, it is good practice to save your plot into a file with a descriptive filename.

1. Building a simple plot

Matplotlib is the most widely used Python library for plotting. Additionally, the plotting functions of Pandas rely on Matplotlib. In order to produce your first plot, we start by importing the Matplotlib module pyplot.

Importing matplotlib main module:

```
import matplotlib.pyplot as plt
```

We then define what we want to plot. In our example, we want to plot the square-root fuction for the x-values from 1 to 100. We define x as a list of the integers from 1 to 100 and y as a NumPy array of their corresponding square-roots.

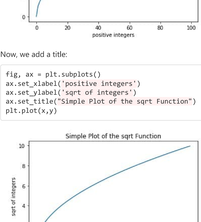
Defining what to plot:

```
import numpy as np

x = [i for i in range(0, 100)]
y = np.sqrt(x)
```

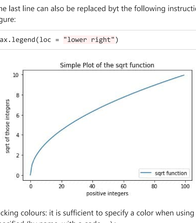
We can now already plot it:

```
plt.plot(x, y)
```



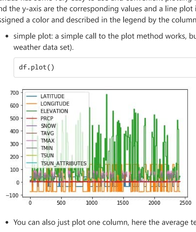
Most of the things we want to add to the plot are features or elements of the axes object. For that, we need to create a "figure" object and an "axes" object. Once we have created the axes object, we can specify the labels/names of the axes:

```
fig, ax = plt.subplots()
ax.set_xlabel('positive integers')
ax.set_ylabel('sqrt of integers')
plt.plot(x,y)
```



Now, we add a title:

```
fig, ax = plt.subplots()
ax.set_xlabel('positive integers')
ax.set_ylabel('sqrt of integers')
ax.set_title("Simple Plot of the sqrt Function")
plt.plot(x,y)
```

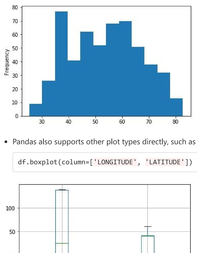


And last but not least we want to add a legend:

```
fig, ax = plt.subplots()
ax.set_xlabel('positive integers')
ax.set_ylabel('sqrt of those integers')
ax.set_title("Simple Plot of the sqrt Function")
ax.plot(x, y, label = "sqrt function")
ax.legend() # This line is necessary to print the legend that will contain our label
```

The last line can also be replaced byt the following instruction, which will plot the legend but also specify its location on the figure:

```
ax.legend(loc = "lower right")
```



Picking colours: it is sufficient to specify a color when using the plot function. Here are all the different colors that can be specified (by name, with a code, ...):

```
ax.plot(x, y, label = "sqrt function", color = "red")
```

Do not forget to actually save your plot into a file: the savefig function allows the creation of an image file containing the figure. The list of available formats is given in the linked documentation. After the last instruction in the previous example, it is sufficient to add the following:

```
fig.savefig("my_fig.png")
```

2. Pandas plotting

Pandas has some very easy to use and convenient plotting features. Without further specification, the x-axis are the row indices and the y-axis are the corresponding values and a line plot is generated. If the whole DataFrame is plotted, each column is assigned a color and described in the legend by the column name.

- simple plot: a simple call to the plot method works, but is most of the time unreadable (here the plot you get with our weather data set).

```
df.plot()
```


- You can also just plot one column, here the average temperature (missing values are left out):

```
df["TAVG"].plot()
```


- As this does not really give us a good overview of the average temperature, we can choose a more appropriate plot type. One useful plot style is the histogram: It groups datapoints in certain ranges together and shows us how many data points are in the respective ranges (so called bins).

```
df["TAVG"].plot.hist(bins=12)
```


- Pandas also supports other plot types directly, such as the in social sciences often used box plots:

```
df.boxplot(column=['LONGITUDE', 'LATITUDE'])
```


- Scatter plots are a very good way to get an overview of discrete data points that are related to each other. For example, if we want to understand how our weather stations are spread over the world according to their coordinates, we can get a quick overview using the scatter plot of `LONGITUDE` and `LATITUDE`. Here we additionally defined the color to be "DarkBlue":

```
df.plot.scatter(x='LONGITUDE',y='LATITUDE', c='DarkBlue')
```