More Matplotlib Functions

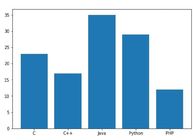
- Often one wants to compare one or several value-sets or functions. To do so, you can use subplots: This means several plots in the same frame. Previously, we used plt.subplots() to create a figure with a single axes-object inside. But we can actually have more than one axes-object. We can also plot different functions on the same figure. The following example uses only one axes-object again but plots two curves:

```
y2 = list(map(lambda x: x**2, x))

fig, ax = plt.subplots()
ax.set_xlabel('positive integers')
ax.set_ylabel('functions values')

ax.set_title("Simple Plot of the Sqrt and Square Functions")
ax.plot(x, y, label = "sqrt function", color="blue")
ax.plot(x, y2, label = "square function", color="red")
ax.legend(loc = "lower right")
```

You can find here an example of multiple subplots.

- Matplotlib gives you a broad variety of possible plotting styles. A wiedly used style is the so called "Bar-Plot" (this is e.g., the standard plot-style pandas uses). Our example is taken from tutorialspoint.com bar plot:

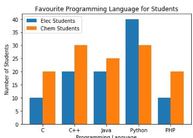
```
fig = plt.figure()
    ax = fig.add_axes([0,0,1,1]) # As we do not have a "natural" x-axis we can add an axes-object like
    langs = ['C', 'C++', 'Java', 'Python', 'PHP']
    students = [23,17,35,29,12]
    ax.bar(langs,students)
    plt.show()
```



A very good example of bar charts is given on the matplotlib site.

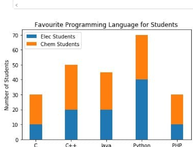
- We often want to compare two or more different groups with respect to different subjects. For example we would like to know if there is a difference in the favourite Programming language between Elec and Chem students. We can do that using a multiple bar plot:

```
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
YElec = [10,20,20,40,10]
ZChem = [20,30,25,30,20]

X_axis = np.arange(len(langs))

plt.bar(X_axis - 0.2, YElec, 0.4, label = 'Elec Students') # Here we place the bar for the Elec Student
plt.bar(X_axis + 0.2, ZChem, 0.4, label = 'Chem Students') # Here we place the bar for the Chem Student

plt.xticks(X_axis, langs)
plt.xlabel("Programming Language")
plt.ylabel("Number of Students")
plt.title("Favourite Programming Language for Students")
plt.legend()
plt.show()
```



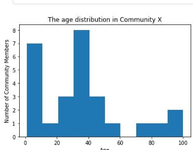
- If we are more intersted in showing the favourite programming language for students but still do not want to loose the information from which field the students chose this language as their favourite a stacked bar plot is a good choice. For that we stack the bars on top of each other. It is called stacked bar chart: https://matplotlib.org/stable/gallery/lines_bars_and_markers/bar_stacked.html

```
langs = ['C', 'C++', 'Java', 'Python', 'PHP'] #my labels()
YElec = [10,20,20,40,10] #Men_means
ZChem = [20,30,25,30,20] #women_means

width = 0.35 # the width of the bars
fig, ax = plt.subplots()
ax.bar(langs, YElec, width, label='Elec Students')
ax.bar(langs, ZChem, width, bottom=YElec, label='Chem Students') # Here we make the bar chart into a st

ax.set_xlabel("Programming Language")
ax.set_ylabel("Number of Students")
ax.set_title("Favourite Programming Language for Students")
ax.legend()

plt.show()
```



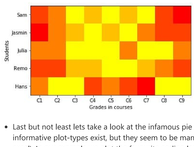
- As already seen with the pandas library we can also directly plot a histogram with matplotlib. If we, e.g., want to take a look at the age distribution of a community, we can plot a histogram of the age of the community members. You can immediately see that there are many youngsters and no people between 60 and 70. histogram:

```
ages = [1,21,22,23,4,5,6,77,8,9,10,34,55,43,89,96,31,32,33,34,35,36,37,18,49,50,100]
num_bins = 10
fig, ax = plt.subplots()
n, bins, patches = plt.hist(ages, num_bins)
ax.set_xlabel("Age")
ax.set_ylabel("Number of Community Members")
ax.set_title("The age distribution in Community X")
plt.show()
```



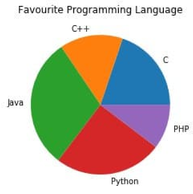
- A really nice way to get a feeling of the combined behaviour of two variables, is to plot a so-called "heatmap". It is a 2D plot that assigns a brighter color to higher values (by default). You can find an overview of the color schemes here: https://matplotlib.org/stable/tutorials/colors/colormaps.html We can use it to get a feeling of the data. In our example we print the grades for 9 different courses (course C1 to C9) from 5 students. You can see that courses C3-C7 seemed easier for most of the students (they got quite a high grade). You can also see that Hans seems to be performing atypical to the majority of the students. An other example is given on the matplot site

```
# The data we want to print is a 2D-Array with grades per student, per course
grades_Sam = [2,3,5,4,5,4,5,2,1]
grades_Jasmin = [1,3,4,5,4,5,3,2,2]
grades_Julia = [3,3,5,5,5,3,5,5,3]
grades_Remo = [2,2,4,4,5,5,4,3,2]
grades_Hans = [3,5,5,2,3,2,1,5,5]
grades = [grades_Sam, grades_Jasmin, grades_Julia, grades_Remo, grades_Hans]
student_Names = ["Sam","Jasmin","Julia","Remo","Hans"]
courses = ["C1","C2","C3","C4","C5","C6","C7","C8","C9"]


fig, ax = plt.subplots()
ax.set_xticks(np.arange(len(courses)))
ax.set_yticks(np.arange(len(student_Names)))
ax.set_xticklabels(courses)
ax.set_yticklabels(student_Names)
ax.set_xlabel("Grades in courses")
ax.set_ylabel("Students")
plt.imshow(grades,cmap='autumn')# cmap is the colorschema used: You can leave it out or choose a differ
plt.show()
```



- Last but not least lets take a look at the infamous pie chart. Pie charts are often used in business. Eventhough more informative plot-types exist, but they seem to be many managers favourites (maybe because they are easy to produce by excel). In our example we plot the favourite coding language from the previous bar plot as a pie plot. If not added additionally one looses a lot of information compared to the bar plot: How many students answered in total? How many students chose a particular programming language etc. On the plus side: Colorful circles look nice. You can find the documentation here pie plot:

```
fig, ax = plt.subplots()
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
plt.pie(students, labels = langs)
ax.set_title("Favourite Programming Language")
plt.show()
```



Here is a guide for the most simple/common plots that can be done with matplotlib. A lot of examples are given with python code and the corresponding figure. Plotting can be quite cumbersome at first. We highly recommend to use code from the documentation and then modify it depending on your needs, and not being afraid to play around with it to see what happens if parameters are changed.