Python Kurs 2019/2020

8: Grafiken

Bernhard Mallinger

`b.mallinger [at] gmx.at`

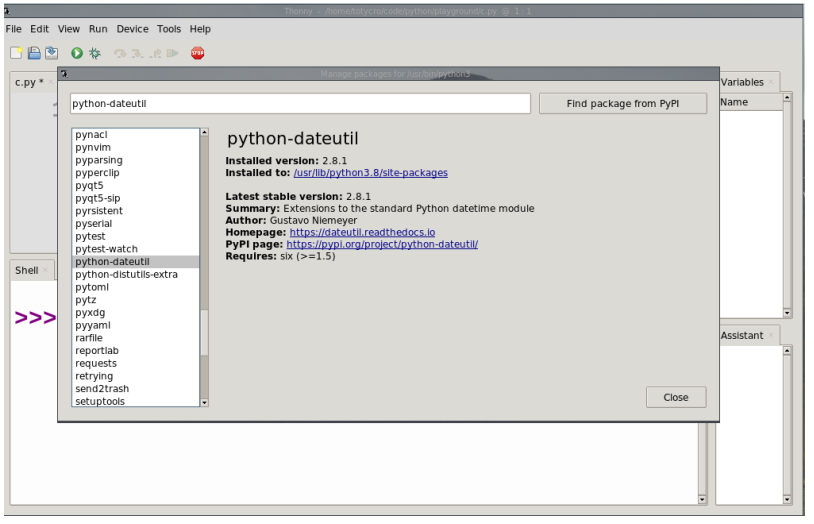
https://totycro.github.io/python-kurs

Exkurs: Paketmanagement

- "batteries included"
- Offizieller Paketmanager: `pip`
- Paketmanager für Wissenschaft: `conda`
- Isolierte Umgebung für Software
  - Offiziell: `venv`
  - Verbreitet in Wissenschaft: `conda` environments
  - Inoffiziell: `virtualenv`, `pyenv`, `pipenv`

Pakete installieren via thonny

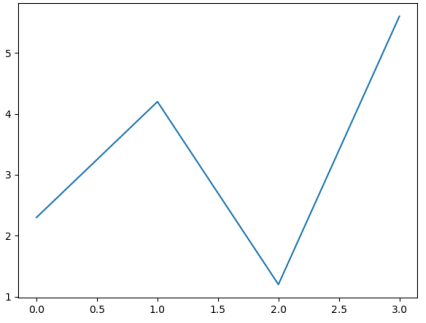
- Interface für `pip`
- Tools → Manage packages
- Installation in Userbereich für alle Projekte eures Users



Plotting: matplotlib

- Teil des Scientific Python Universums für Grafiken
- z.B. über Thonny installieren
- "Matplotlib tries to make easy things easy and hard things possible."

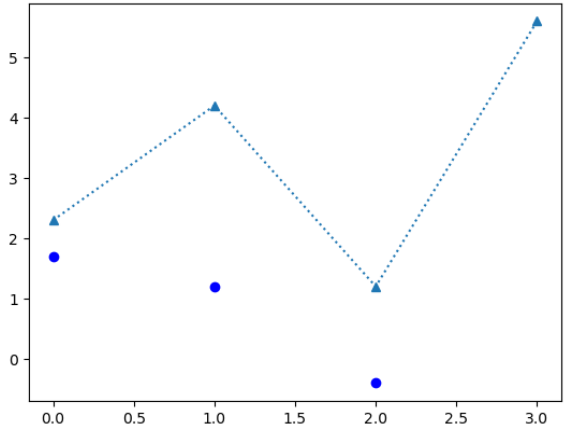
```
       matplotlib.pyplot    plt
plt.plot([2.3, 4.2, 1.2, 5.6])
plt.show()
```



Plotting

- Hier: pyplot function interface von matplotlib
- Erlaubt `fmt` (format) (Beschreibung hier unter "Notes")

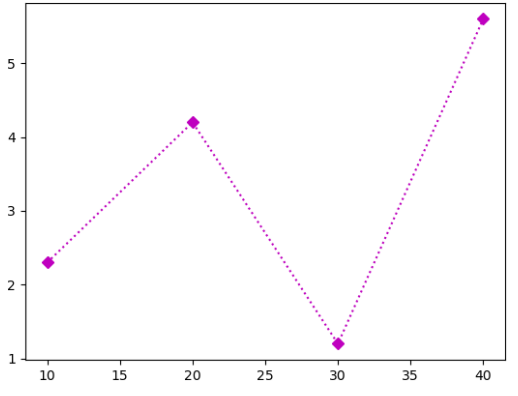
```
plt.plot([2.3, 4.2, 1.2, 5.6], "^:")
plt.plot([1.7, 1.2, -0.4], "ob")
```



Plotting

- X-Werte können auch explizit angegeben werden
- Achtung: Keyword-Arguments kaputt, es wird "geraten", was `x`, `y` und `fmt` ist

```
plt.plot([10, 20, 30, 40], [2.3, 4.2, 1.2, 5.6], "D:m")
```



Plotting: Zusätzliche Parameter

Titel definieren:
`plt.title('My Title')`

Text schreiben:
`plt.text(4, 3, 'Hier ist ein wichtiger Punkt')`

Annotation:
`plt.annotate("Hier", xy=(2.5, 6), xytext=(3.5, 5), arrowprops={'facecolor': 'red'})`

Achsenbeschreibung:
`plt.xlabel("X-Achse")`, `plt.ylabel("Y-Achse")`

Plotting: Zusätzliche Parameter

Legende:
`plt.legend(["Kurve 1", "Kurve 2"])`

Wertebereich definieren:
`plt.axis([0, 10, 0, 20])`

Flächen zeichen:
`plt.fill_between(x=range(len(values1)), y1=values1, y2=values2, alpha=0.3)`

Grafiken speichern statt anzeigen:
`plt.savefig("grafik.png") # anstatt: plt.show()`

Histogramme

```
plt.hist([1,3,2,1,2,4,3,2,3,4,5,1,2,3,2,2], bins=5)
```

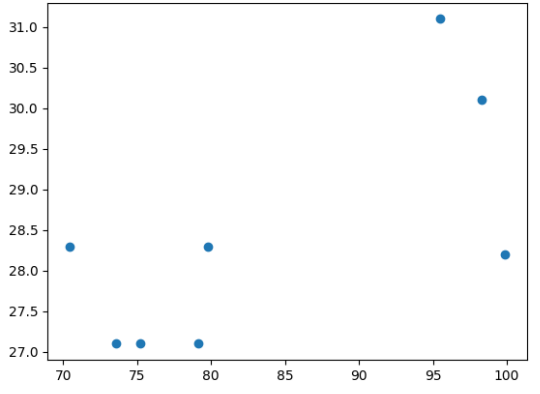
Weitere interessante Parameter:

- `density=True`: Y-Achse ist wird normalisiert, so dass Summe 1 ist
- `cumulative=True`: Kumulatives Diagramm
- `rwidth=0.85`: Breite der Balken (Wert zwischen 0 und 1)

Streudiagramm

```
# Daten von Produktionsreihen
ausbeute = [99.9, 70.4, 79.8, 75.2, 95.5, 79.1, 73.6, 98.3]
temperatur = [28.2, 28.3, 28.3, 27.1, 31.1, 27.1, 27.1, 30.1]

plt.scatter(x=ausbeute, y=temperatur)
```



Streudiagramm

Mit Farben, Größen und Achsenbeschriftung:

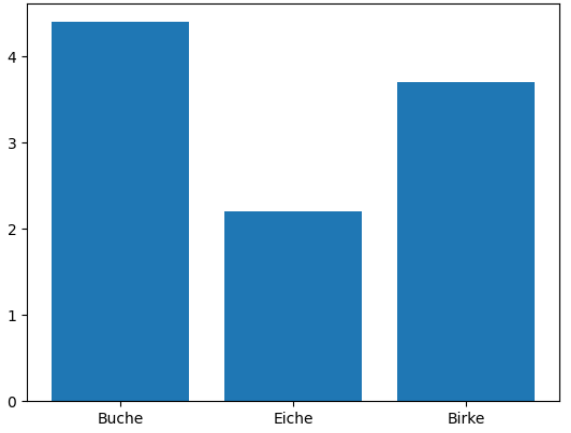
```
ausbeute = [99.9, 70.4, 79.8, 75.2, 95.5, 79.1, 73.6, 98.3]
temperatur = [28.2, 28.3, 28.3, 27.1, 31.1, 27.1, 27.1, 30.1]

        (value):
      value >= 80:
              "g"
       :
              "b"

plt.scatter(
    x=ausbeute,
    y=temperatur,
    c=[color(val)     val    ausbeute],
    s=[(val - 60) ** 2     val    ausbeute],
)
plt.xlabel("Ausbeute")
plt.ylabel("Temperatur")
```

Säulendiagramm

```
tree_growth = {
    "Buche": 4.4,
    "Eiche": 2.2,
    "Birke": 3.7,
}
plt.bar(x=tree_growth.keys(), height=tree_growth.values())
```



Säulendiagram: Von COVID-19 betroffene Länder

```
     urllib.request        urlopen, Request
       json
       matplotlib.pyplot    plt

            ():
          json.loads(urlopen(Request(
        "https://corona.lmao.ninja/countries?sort=cases",
        headers={"User-Agent": "Mozilla/5.0"},
    )).read())

country_cases = fetch_data()
# country_cases is sorted list of dicts with country case data

top10 = country_cases[:10]
plt.bar(
    x=[country_data["country"]     country_data    top10],
    height=[country_data["cases"]     country_data    top10],
)
plt.show()
```

Weitere Diagrammtypen

```
plt.boxplot([1,3,2,1,2,4,3,2,3,4,5,1,2,3,2,2])
```

```
plt.pie([5, 3, 7], labels=["Birke", "Buche", "Eiche"])
```

Plotting

AUFGABE

Generiere eine Grafik aus deinem Arbeitsbereich mittels `matplotlib`.

Plotting: Weiterführende Links

`help(plt.hist)`

Offizielle PyPlot Einführung: Viele weitere Beispiele und Tutorials, aber oft sehr advanced!

https://towardsdatascience.com/matplotlib-tutorial-learn-basics-of-pythons-powerful-plotting-library-b5d1b8f67596

https://www.python-kurs.eu/matplotlib.php

Seaborn

https://seaborn.pydata.org/introduction.html

'If matplotlib "tries to make easy things easy and hard things possible", seaborn tries to make a well-defined set of hard things easy too.'