# Matplotlib und Seaborn

Mathplotlib (https://matplotlib.org/) ist eine Sammlung von Funktionen (Bibliothek) zum Visualisieren von Daten. Wir verwenden mathplotlib zusammen mit dem ergänzenden Programmpaket Seaborn.

Wichtig: Sie müssen die folgenden beiden Zeilen stets am Beginn des Programms stehen haben.

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

# Plots

## Scatterplot

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

years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [33.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3]

sns.set()
fig,ax = plt.subplots(figsize=(9, 9))
ax.set_title("Title")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")
#ax.set_aspect('equal')
#ax.set_xlim(0, 50)
#ax.set_ylim(0, 35)

sns.scatterplot(x=years, y=gdp, color="red", label="My Label")
```

Hinweise:

- Zu Farben siehe : https://seaborn.pydata.org/tutorial/color_palettes.html (Vorsicht, anspruchsvoll)
- Formatierung der Achsen beachten

## Liniendiagramme

Hier ein erstes Beispiel:

```
#Line
sns.scatterplot(x=years, y=gdp, color="red", label="My Label")
```

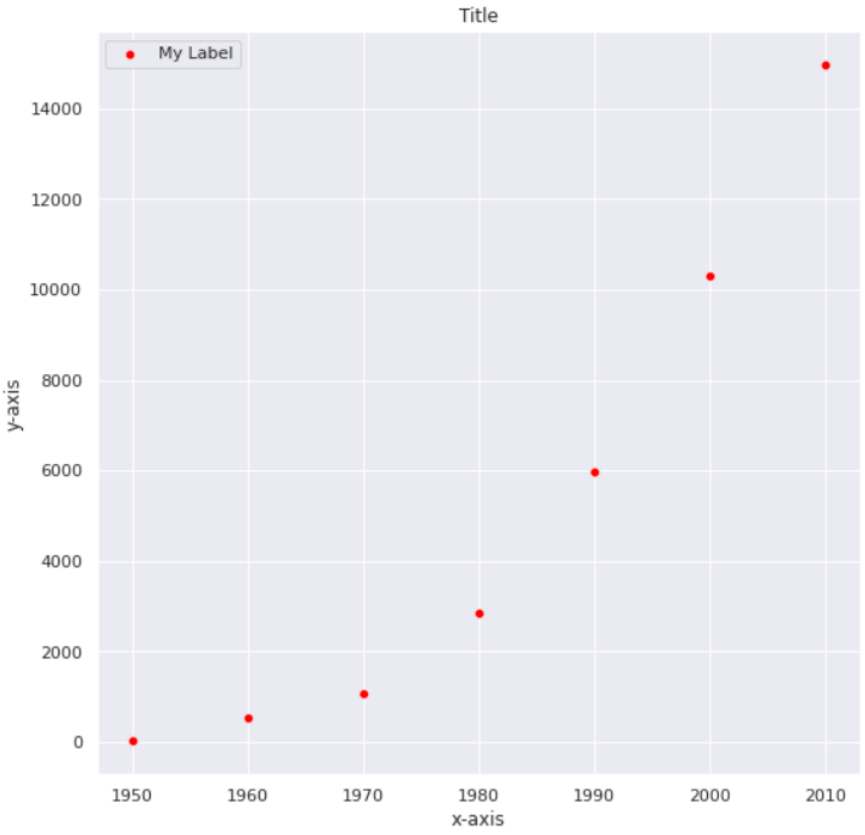


Figure 1: Ausgabe

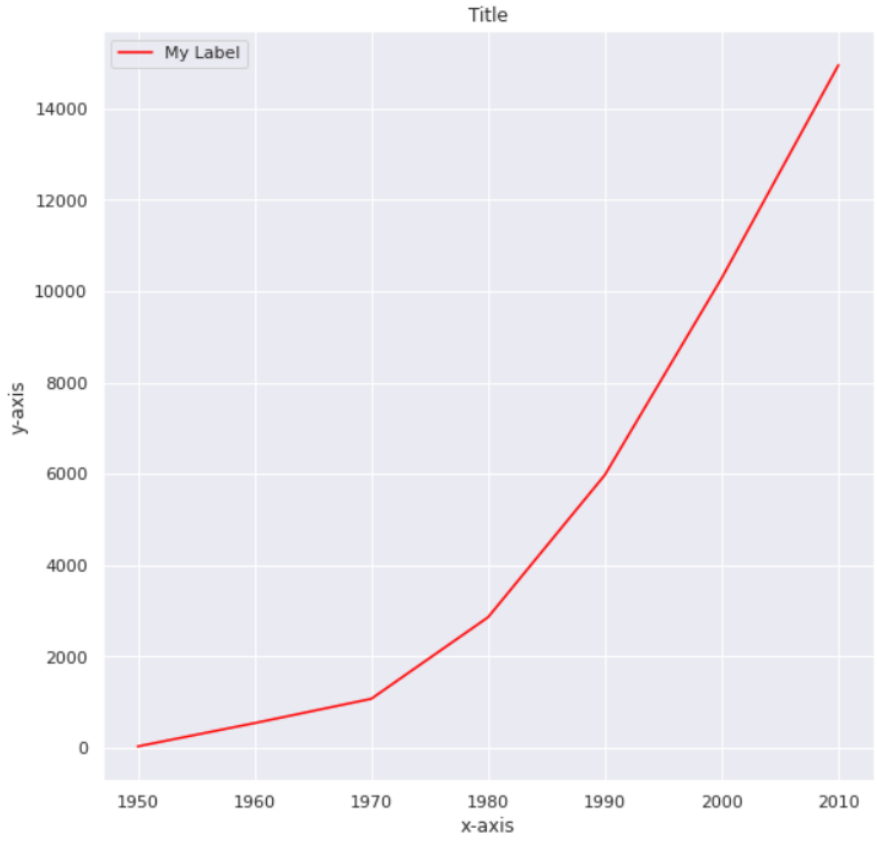


Figure 2: image-20211209164634964

## Barplot

```
sns.barplot(x=years, y=gdp, color="red", label="My Label")
```

## Histogramme

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

x_werte =  [1, 2, 2, 3,3, 4, 5, 6, 7, 8, 9, 10]

sns.set()
sns.histplot(x = x_werte,
             #binwidth=1,
             #bins="auto",
             bins=[0,2,5,7,10],
             kde = False)
```

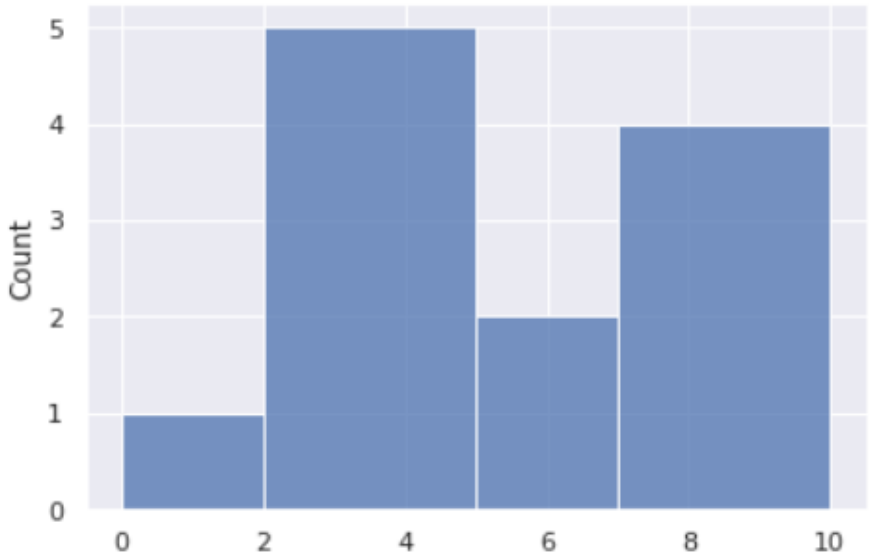


Figure 3: image-20211210105040430

## Boxplots

Link: https://seaborn.pydata.org/generated/seaborn.boxplot.html

Link: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

Vertiefung:

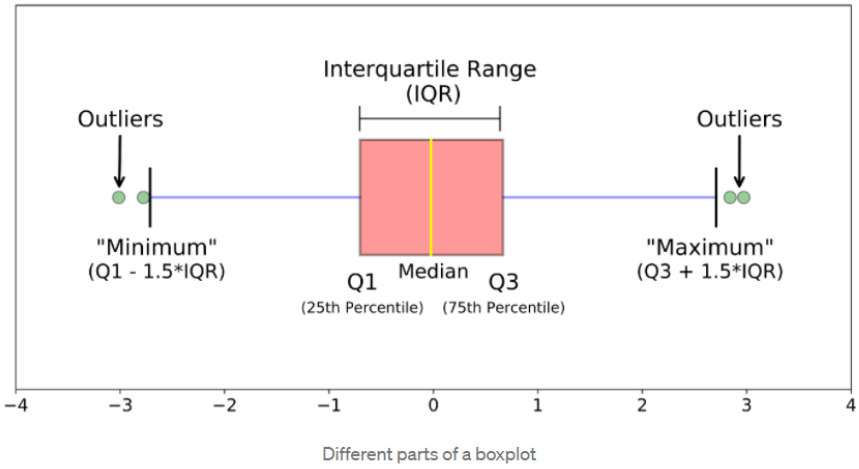


Figure 4: image-20211210123539036

Outliers, Inliers, and Other Surprises that Fly from your Data | Rocket-Powered Data Science (rocketdatascience.org)

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

x = np.arange(1,101)
x = np.concatenate( (x, [152] ))

percentiles = np.percentile( x, [0,25,50,75,100])
IQR = (percentiles[3] -  percentiles[1])
print( "Pericentile         : ", percentiles)
print( "IQR                 : ", IQR)
print( "Upper Outlier Limit : ", percentiles[3] + 1.5*IQR)
sns.boxplot(x = x)
```

```
Pericentile         :  [  1.  26.  51.  76. 152.]
IQR                 :  50.0
Upper Outlier Limit :  151.0
```

Figure 5: image-20211210124842057

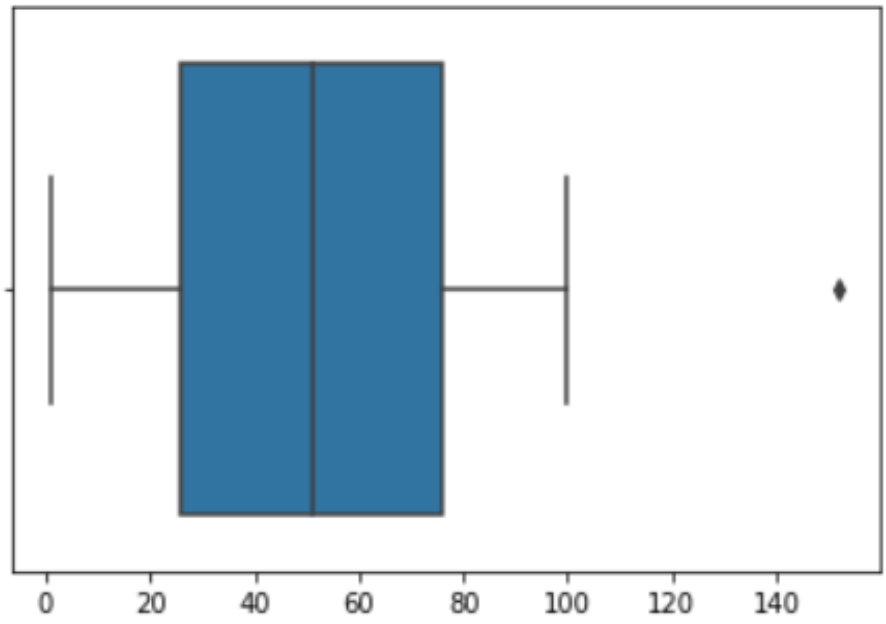


Figure 6: image-20211210124819306

# Übungen

## Visualisierungsübung 1

Versuchen sie folgende Kurve zu zeichnen:

(Die Lösung finden sie am Ende des Dokumentes)

## Visualisierungsübung 2

1. Sie können mit numpy eine Liste mit 50 gleichverteilen Zufallszahlen erstellen. Erzeugen Sie zwei dieser Listen (x und y) und zeigen sie die 50 Paare Paare (x[i], y[i]) in einem Scatterplot an. So ähnlich (!) sollte die Ausgabe aussehen:
2. Welches Ausgabe erwarten Sie, wenn Sie in 1. statt 50 Zahlen jeweils 2000 Zahlen erzeugen? Beschreiben sie das Ergebnis in 2-3 Sätzen.
3. Wiederholen sie Aufgabe 1 mit der Normalverteilung (Erwartungswert 0, Standardabweichung 1) statt der Gleichverteilung. Verwenden sie 2000 Punkte. Überlegen sie bitte vorher: welche grafischen Ausgabe erwarten sie?
4. Erzeugen Sie ein Histogramm für die Erzeugung von 10000 gleichverteilten

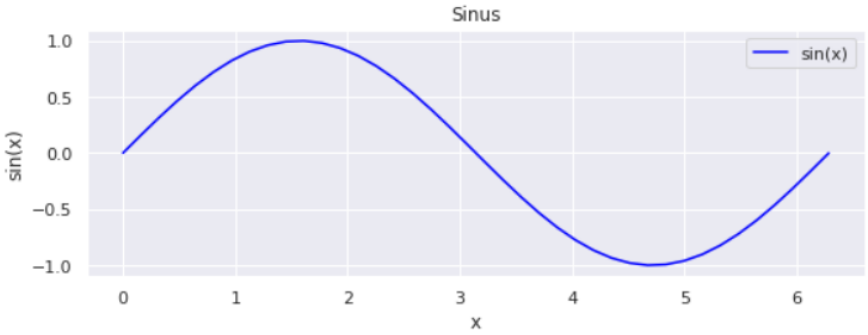


Figure 7: image-20211210101947217

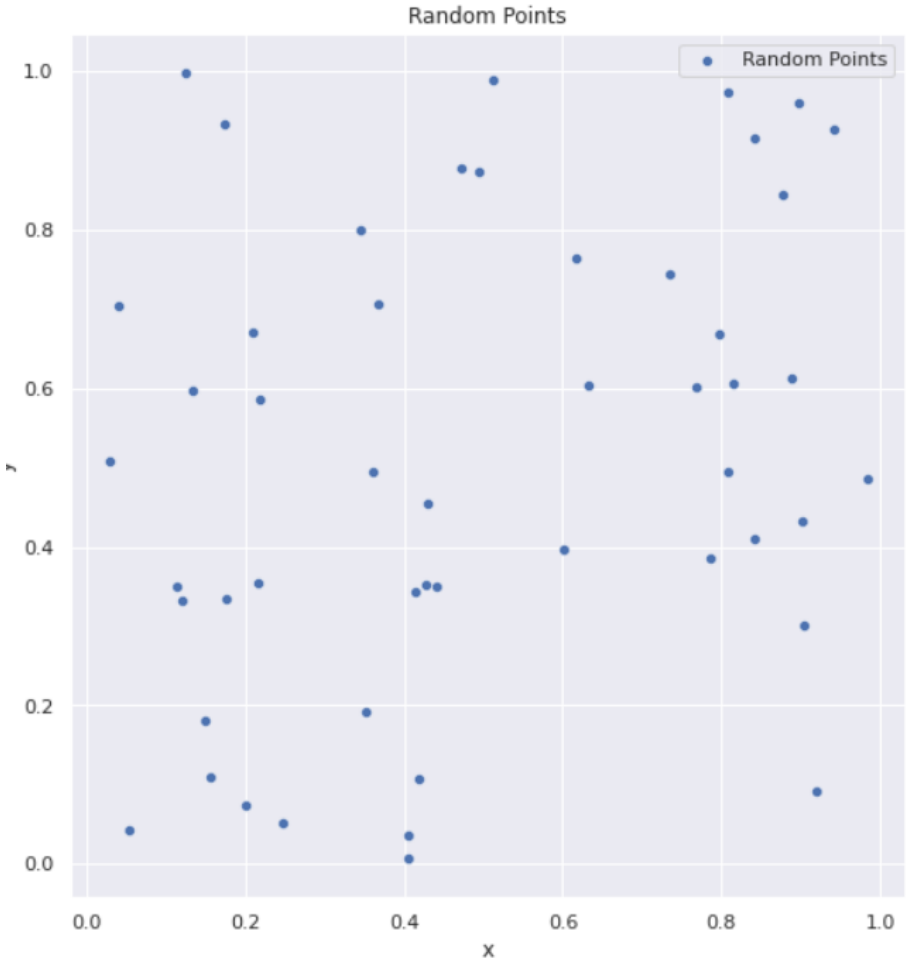


Figure 8: image-20211210102949673

(normalverteilten) Zufallszahlen. So ähnlich sollte das Ergebnis aussehen:

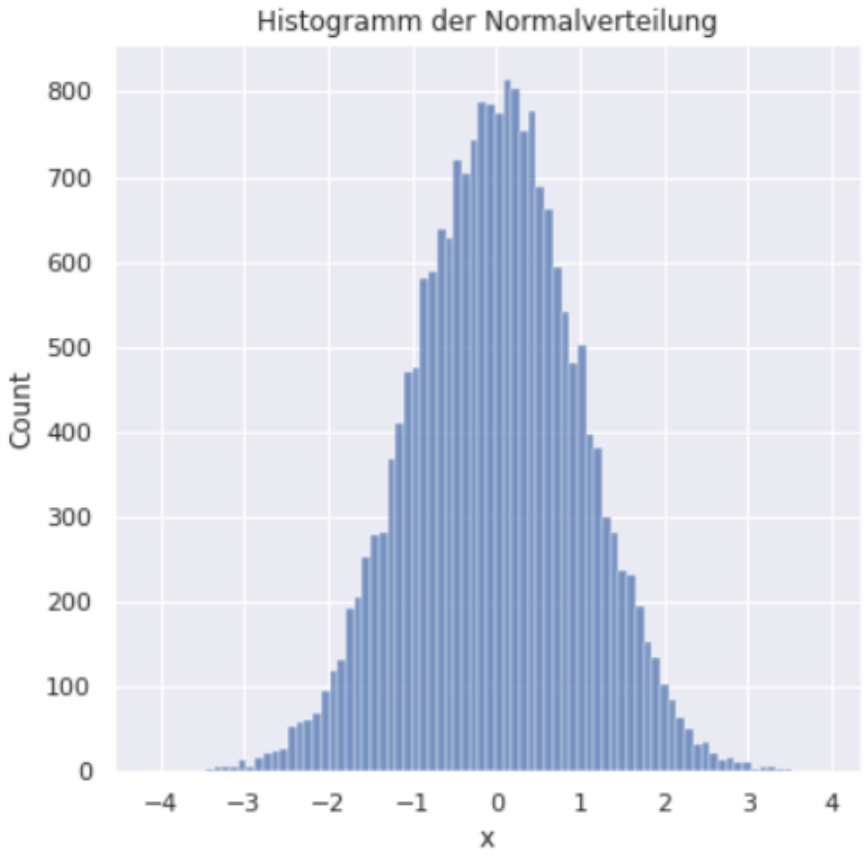


Figure 9: image-20211210103950590

# Plots für Iris

## Laden der Daten

```
import pandas as pd
from sklearn import datasets

import matplotlib.pyplot as plt
import seaborn as sns

iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data)
iris_df['class']=iris.target_names[iris.target ]
```

```
iris_df.columns=['sepal_len', 'sepal_wid', 'petal_len', 'petal_wid', 'class']
```

### Histogramm für ein Feature

```
sns.set()
sns.histplot( iris_df,
             x ="petal_wid",
             bins=10,  # try bins="auto" und bins=10
             kde = False)
```

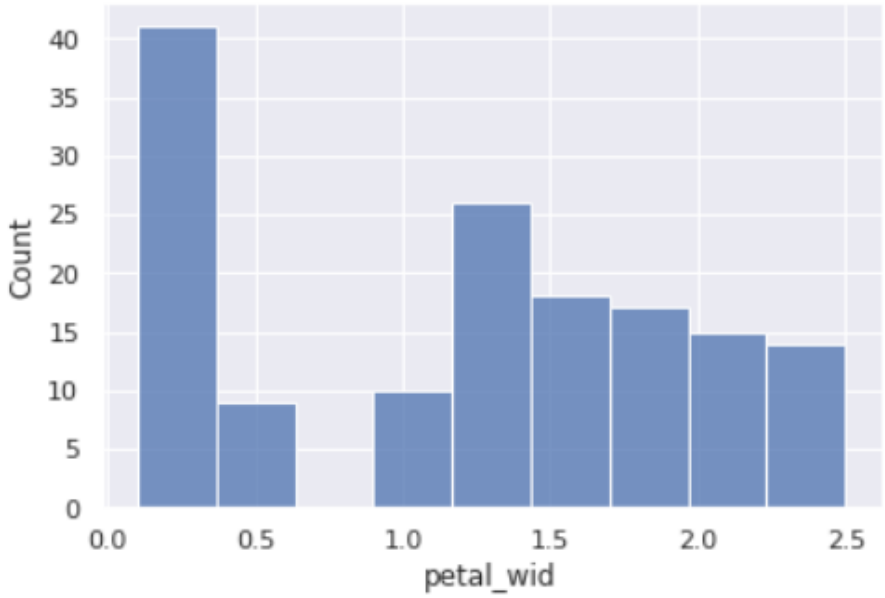


Figure 10: image-20211209170610859

### Relation-Plots

```
sns.relplot(data=iris_df, x="petal_len", y="sepal_wid", hue = "class", height=7)
```

### Pairplots

```
sns.pairplot(iris_df[['sepal_len', 'sepal_wid', 'petal_len', 'petal_wid', 'class']],
             hue="class", diag_kind="kde")
```

Ausgabe:

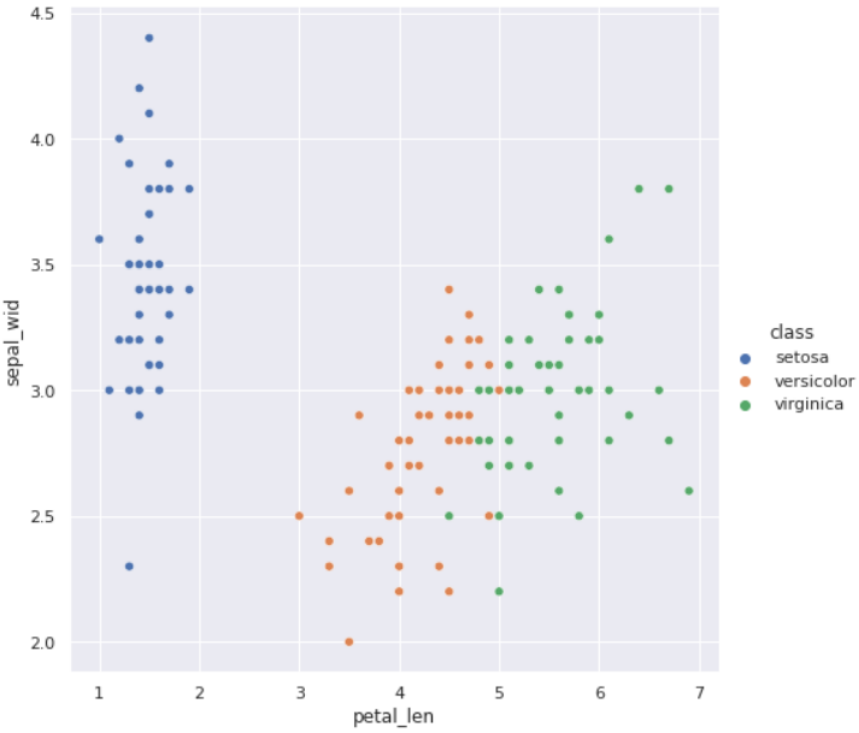


Figure 11: image-20211209170740088

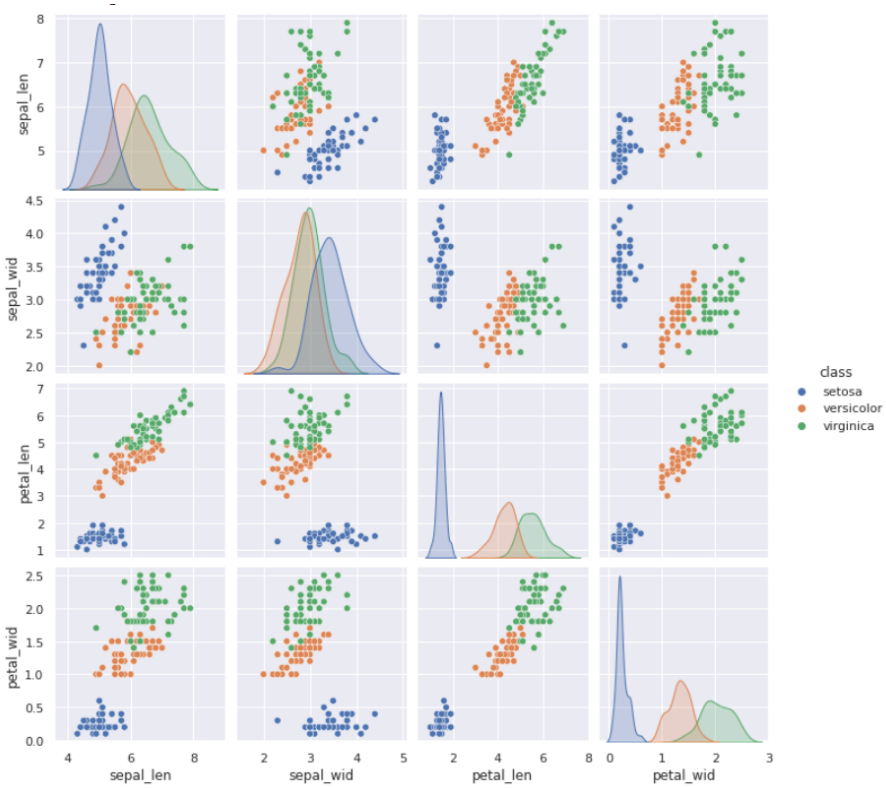


Figure 12: image-20211209171426965

## Beispiele: Bilder

Bilder lassen sich als Numpy-Arrays darstellen und bearbeiten. Unterstützte shapes sind:

- (M, N): ein Bild mit skalaren Werten (0-1 float oder 0-255 int). Visualisierung erfolgt über eine *colormap*.
- (M, N, 3): ein Farbbild mit RGB Werten (0-1 float oder 0-255 int).

Die ersten beiden Werte (M, N) definieren die Anzahl der Zeilen und Spalten des Bildes.

(Taken from https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.imshow.html)

### Grauwert-Bilder als nxm Matrix

Folgende Befehle erzeugen ein künstliches und (gleichverteilt) zufälliges Grauwertbild:

```
import matplotlib.pyplot as plt
import numpy as np

img = np.random.rand(10,10)
plt.imshow(img, cmap= plt.cm.get_cmap('Oranges'), vmin=0, vmax=1  )
```

Colormaps finden sie unter https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html. Sie können auch probieren: 'Greys', 'Purples', 'Blues', 'Greens', 'Oranges', 'Reds', 'YlOrBr', 'YlOrRd', 'OrRd', 'PuRd', 'RdPu', 'BuPu', 'GnBu', 'PuBu', 'YlGnBu', 'PuBuGn', 'BuGn', 'YlGn'

### Farbbilder aus Datei laden

```
import numpy as np
import matplotlib.pyplot as plt

url = 'http://www.dietergreipl.de/wp-content/uploads/2019/10/owl-50267_1920.png'
eule = plt.imread( url )
print(eule.shape)
print(np.amax( eule ))
print(np.amin( eule ))
plt.figure( figsize=(20,15))
plt.imshow( eule )
```

### Einfache Farbbilder erzeugen

```
import numpy as np
import matplotlib.pyplot as plt
img = np.zeros( (200,200,3))
```


Figure 13: image-20211210112329871

```
img[:,:,1] = np.ones((200))
plt.figure()
plt.imshow( img  )
```

### Komplett zufälliges Farbbild

```
import numpy as np
import matplotlib.pyplot as plt
img = np.random.random( 200*200*3).reshape(200,200,3)
plt.figure(figsize= (9,9))
plt.imshow( img  )
```

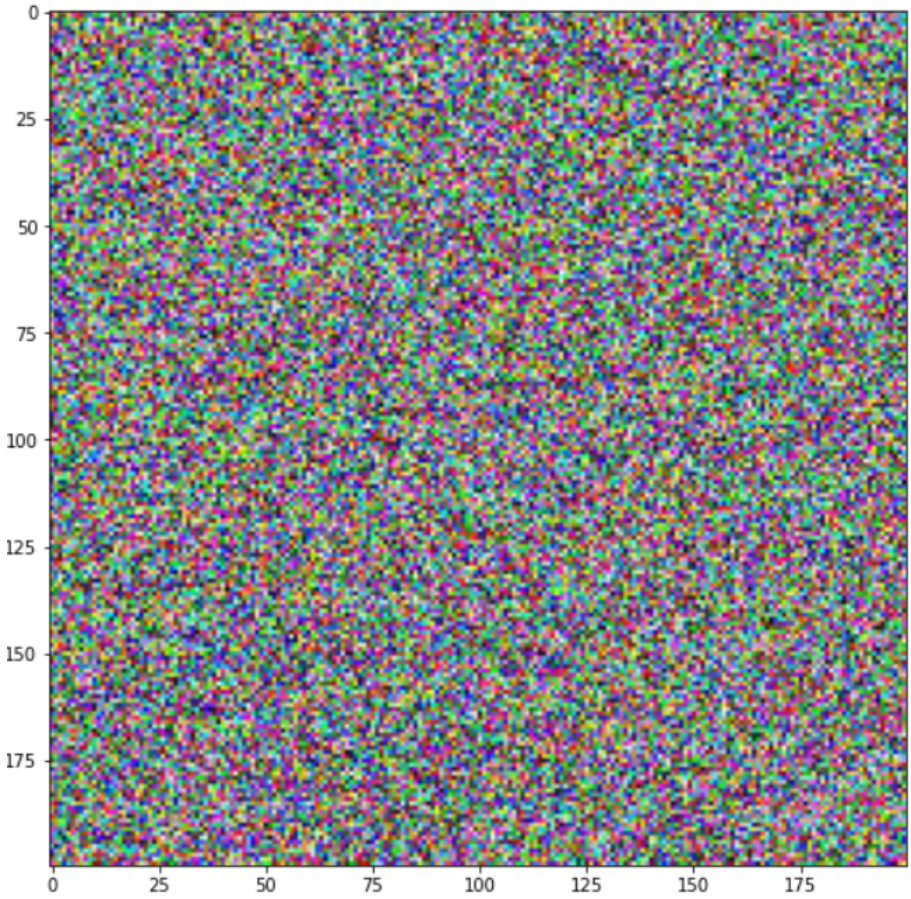


Figure 14: image-20211210112144985

### Übungen

**Wie kommt dieses Bild zustande?** Erläutern Sie, wie das Bild erstellt wird, speziell die for-Schleife:

```
import numpy as np
import matplotlib.pyplot as plt

img= np.ones((200, 200))

for col in range(0, 200):
  img[:,col] = col

plt.figure(figsize= (9,9))
plt.imshow( img, cmap= plt.get_cmap('gray'), vmin=0, vmax=200)
```

Ausgabe:

**Erklären sie die Ausgabe dieses Programms:**

```
import numpy as np
import matplotlib.pyplot as plt
img = np.random.normal(0.5, 0.1, 200*200*3).reshape(200,200,3)
plt.figure(figsize= (9,9))
plt.imshow( img  )
```

Ausgabe:

# Sandbox

## Alle Farben

Folgendes Programm gibt alle Farben mit Farbnamen aus:

```
from matplotlib.patches import Rectangle
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors


def plot_colortable(colors, title, sort_colors=True, emptycols=0):

    cell_width = 212
    cell_height = 22
    swatch_width = 48
    margin = 12
    topmargin = 40

    # Sort colors by hue, saturation, value and name.
```

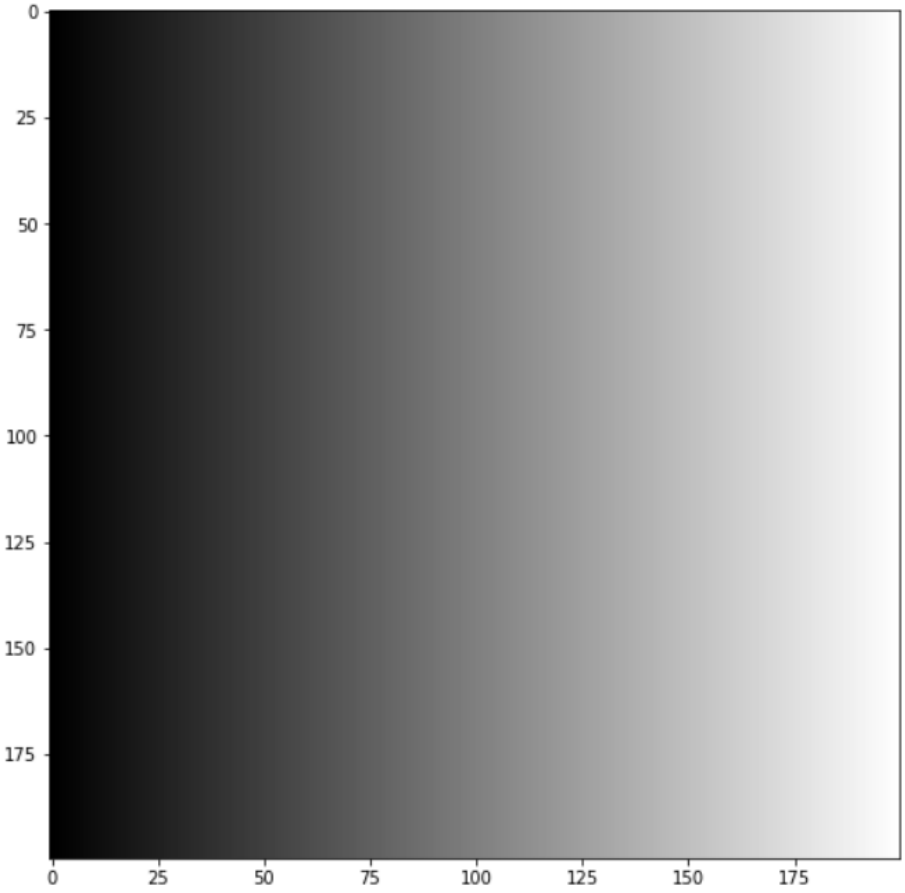


Figure 15: image-20211210112001756

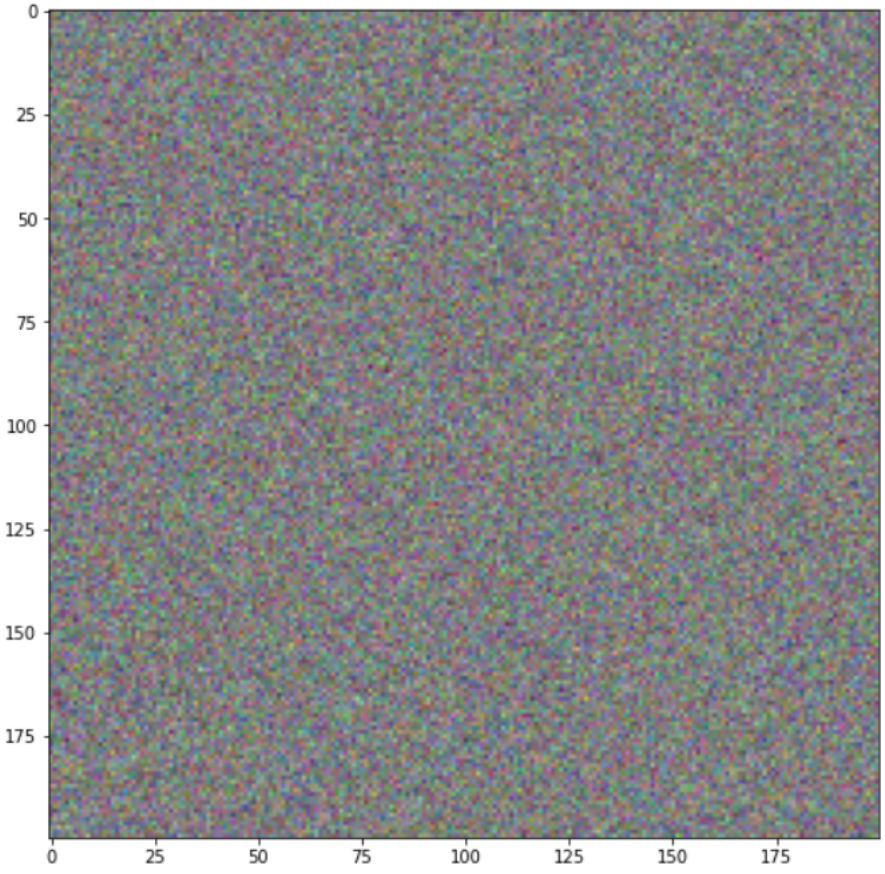


Figure 16: image-20211210113514004

```
    if sort_colors is True:
        by_hsv = sorted((tuple(mcolors.rgb_to_hsv(mcolors.to_rgb(color))),
                         name)
                        for name, color in colors.items())
        names = [name for hsv, name in by_hsv]
    else:
        names = list(colors)

    n = len(names)
    ncols = 4 - emptycols
    nrows = n // ncols + int(n % ncols > 0)

    width = cell_width * 4 + 2 * margin
    height = cell_height * nrows + margin + topmargin
    dpi = 72

    fig, ax = plt.subplots(figsize=(width / dpi, height / dpi), dpi=dpi)
    fig.subplots_adjust(margin/width, margin/height,
                        (width-margin)/width, (height-topmargin)/height)
    ax.set_xlim(0, cell_width * 4)
    ax.set_ylim(cell_height * (nrows-0.5), -cell_height/2.)
    ax.yaxis.set_visible(False)
    ax.xaxis.set_visible(False)
    ax.set_axis_off()
    ax.set_title(title, fontsize=24, loc="left", pad=10)

    for i, name in enumerate(names):
        row = i % nrows
        col = i // nrows
        y = row * cell_height

        swatch_start_x = cell_width * col
        text_pos_x = cell_width * col + swatch_width + 7

        ax.text(text_pos_x, y, name, fontsize=14,
                horizontalalignment='left',
                verticalalignment='center')

        ax.add_patch(
            Rectangle(xy=(swatch_start_x, y-9), width=swatch_width,
                      height=18, facecolor=colors[name], edgecolor='0.7')
        )

    return fig

plot_colortable(mcolors.BASE_COLORS, "Base Colors",
```

```
                sort_colors=False, emptycols=1)
plot_colortable(mcolors.TABLEAU_COLORS, "Tableau Palette",
                sort_colors=False, emptycols=2)

plot_colortable(mcolors.CSS4_COLORS, "CSS Colors")

# Optionally plot the XKCD colors (Caution: will produce large figure)
# xkcd_fig = plot_colortable(mcolors.XKCD_COLORS, "XKCD Colors")
# xkcd_fig.savefig("XKCD_Colors.png")

plt.show()
```

## Lösungen

### Visualisierung 1

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

x = np.linspace(0, 2*np.pi, 40)

fig,ax = plt.subplots(figsize=(9, 9))

ax.set_title("Sinus")
ax.set_xlabel("x")
ax.set_ylabel("sin(x)")
ax.set_aspect("equal")

sns.set()
sns.lineplot(x=x, y= np.sin(x), color="blue", label="sin(x)")
```

### Visualisierung 2

#### 2.1

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

x = np.random.random(50)
y = np.random.random(50)

fig,ax = plt.subplots(figsize=(9, 9))

sns.set()
ax.set_title("Random Points")
```

```
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_aspect("equal")

sns.scatterplot(x=x, y=y, label="Random Points")
```

**2.3**

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

x = np.random.normal(0,1,2000)
y = np.random.normal(0,1,2000)

fig,ax = plt.subplots(figsize=(9, 9))

sns.set()
ax.set_title("Random Points")
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_aspect("equal")

sns.scatterplot(x=x, y=y, label="Random Points")
```

**2.4**

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

x = np.random.normal(0,1,20000)


fig,ax = plt.subplots(figsize=(6, 6))

sns.set()
ax.set_title("Histogramm der Normalverteilung")
ax.set_xlabel("x")
ax.set_ylabel("Count")

sns.histplot(x=x)
```