`matplotlib` – mehr als nur eine 2D Diagramm Bibliothek

Bei `matplotlib` handelt es sich um eine Python 2D Diagramm Bibliothek – wir können also sehr einfach grafische Darstellungen von Daten und Zahlenreihen erstellen.

Zahlreiche Beispiele finden sich auf der dazugehörigen Website unter https://matplotlib.org

Im ersten Schritt müssen wir die Bibliothek über PIP installieren:

```
pip install matplotlib
```

Nach der Installation können wir die Bibliothek durch den typischen Import in Python nutzen:

```
import matplotlib as plt
```

Hier wird die Abkürzung „plt“ genutzt – Konvention hilft anderen Programmieren schneller Python-Code zu verstehen, daher sollte man solche Konventionen einhalten.

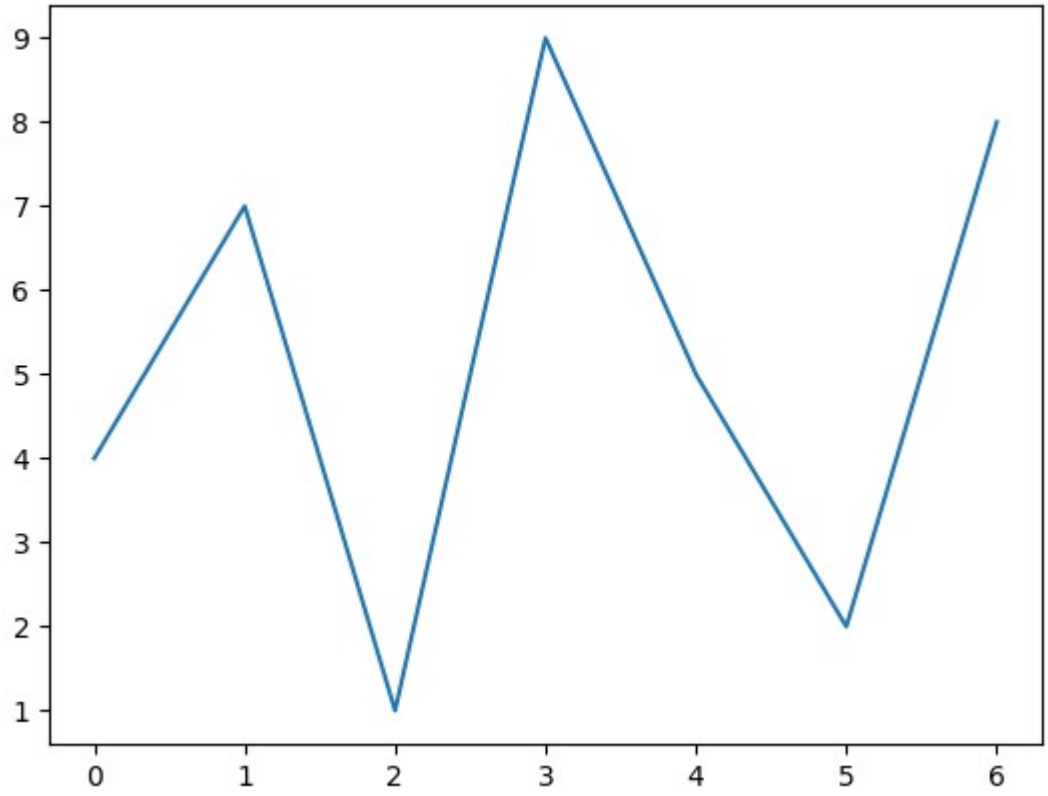
Wir erstellen nun eine Liste mit Zahlen, die wir als Diagramm ausgeben lassen wollen:

```
import matplotlib.pyplot as plt

daten = [4, 7, 1, 9, 5, 2, 8]
```

Jetzt müssen wir die Form der Ausgabe festlegen über `plot()` und die Ausgabe starten über `show()`:

```
import matplotlib.pyplot as plt

daten = [4, 7, 1, 9, 5, 2, 8]
plt.plot(daten)
plt.show()
```

Wir erhalten nun in einem neuen Fenster die Ausgabe:

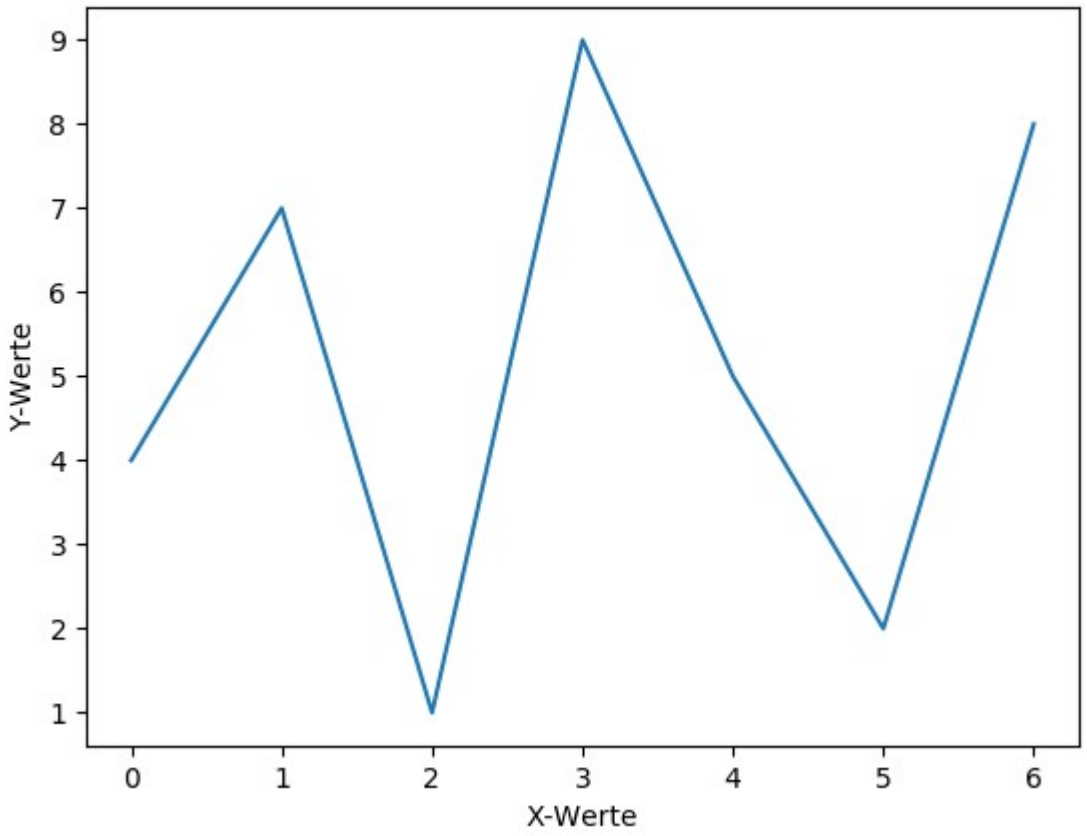


Beschriftung der Achsen aktivieren

Um eine Beschriftung der Achsen zu aktivieren, geben wir die Anweisung `plt.xlabel()` und `plt.ylabel()` mit. Dies muss vor der Ausgabe mit `show()` erfolgen:

```
import matplotlib.pyplot as plt

daten = [4, 7, 1, 9, 5, 2, 8]
plt.plot(daten)
plt.xlabel("X-Werte")
plt.ylabel("Y-Werte")
plt.show()
```

Als Ergebnis erhalten wir die Beschriftung unter der x-Achse und links neben der y-Achse.



Ausgabe als Balkendiagramm

Bemerkenswert ist, dass die Werte der x-Achse einfach erstellt werden. Bei vielen Diagramme benötigen wir sowohl die X wie Y-Werte. Wir machen aus unserer Liste mit der Bezeichnung `daten` nun unsere Liste `ywerte` und erstellen eine zweite Liste `xwerte`. Diese könnten wir auch über `range()` erstellen. Als Ausgabe wählen wir den Diagrammtyp `plt.bar(x, y)` und übergeben die beiden Listen:

```
import matplotlib.pyplot as plt

ywerte = [4, 7, 1, 9, 5, 2, 8]
xwerte = [1, 2, 3, 4, 5, 6, 7]
plt.bar(xwerte, ywerte)
plt.xlabel("X-Werte")
plt.ylabel("Y-Werte")
plt.show()
```

Wir erhalten nun ein Balkendiagramm:

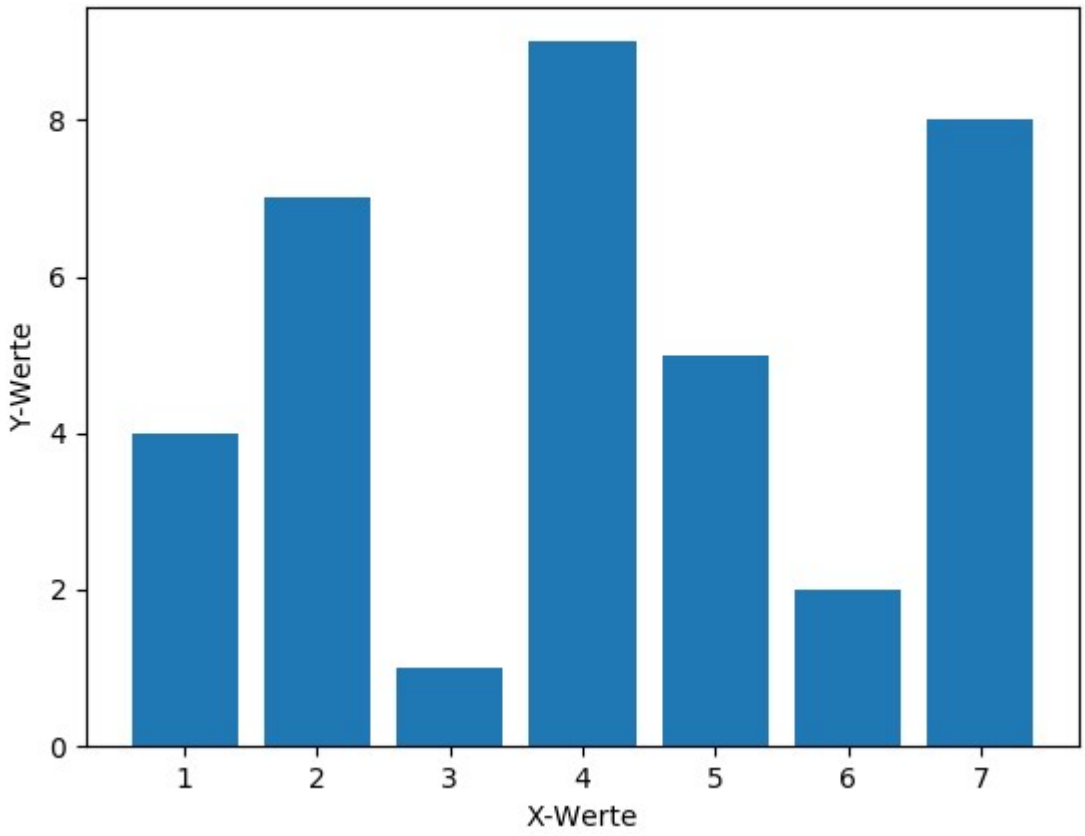
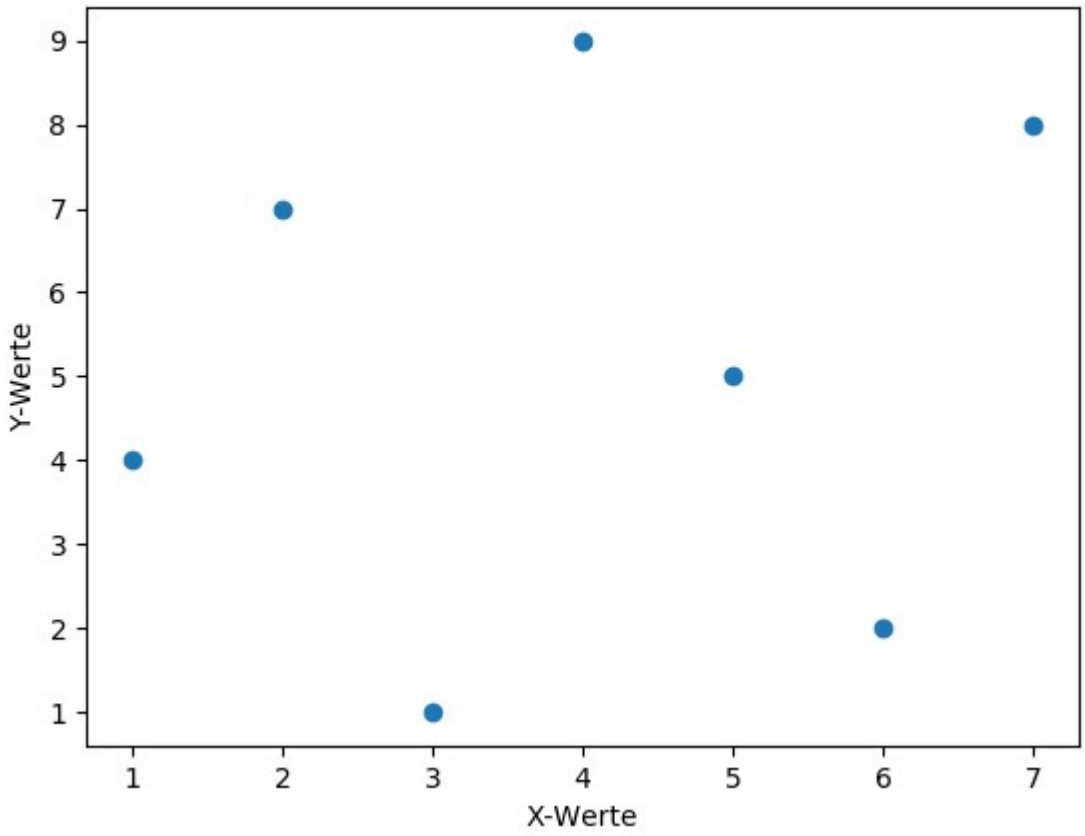


Diagramm mit Punkten

Diagrammtyp `plt.scatter(x, y)` auswählen und die beiden Listen übergeben:

```
import matplotlib.pyplot as plt

ywerte = [4, 7, 1, 9, 5, 2, 8]
xwerte = [1, 2, 3, 4, 5, 6, 7]
plt.scatter(xwerte, ywerte)
plt.xlabel("X-Werte")
plt.ylabel("Y-Werte")
plt.show()
```

Wir erhalten nun ein Diagramm mit Punkten:



Mehrere Diagramme kombinieren

Kombiniert man mehrere Diagrammtypen wie beispielsweise das Liniendiagramm mit dem Punktdiagramm, kann ein Mehrwert entstehen:

```
import matplotlib.pyplot as plt

ywerte = [4, 7, 1, 9, 5, 2, 8]
xwerte = [1, 2, 3, 4, 5, 6, 7]
plt.plot(xwerte, ywerte)
plt.scatter(xwerte, ywerte)
plt.xlabel("X-Werte")
plt.ylabel("Y-Werte")
plt.show()
```

Möchte man nun die Punkte hervorheben und diesen eine andere Farbe zuweisen, ist das kein Problem. Die Anweisung `plt.scatter(x,y,c=color)` wird um den Parameter „c“ für Farbe (color) erweitert:

```
plt.scatter(xwerte, ywerte, color='red')
```

Einfach testen – als Ergebnis sollte folgende Diagrammausgabe erscheinen:

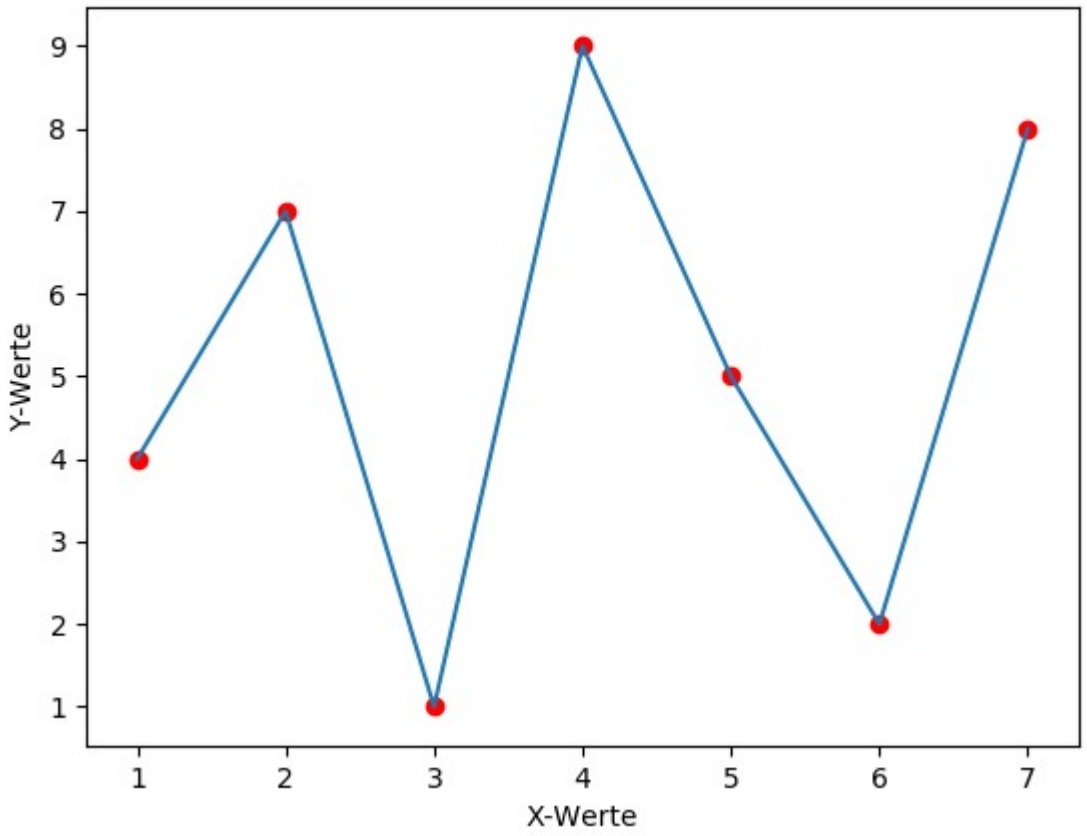


Diagramm als Grafik speichern

Neben der Ausgabe eines Diagramms beherrscht die matplotlib-Bibliothek auch das Speichern der Diagramme als Datei. Dafür gibt es die Anweisung:

```
plt.savefig('gespeichertesdiagramm.png')
```

Diese kann anstelle von `plt.show()` oder zusätzlich zu der Bildschirmausgabe erfolgen.

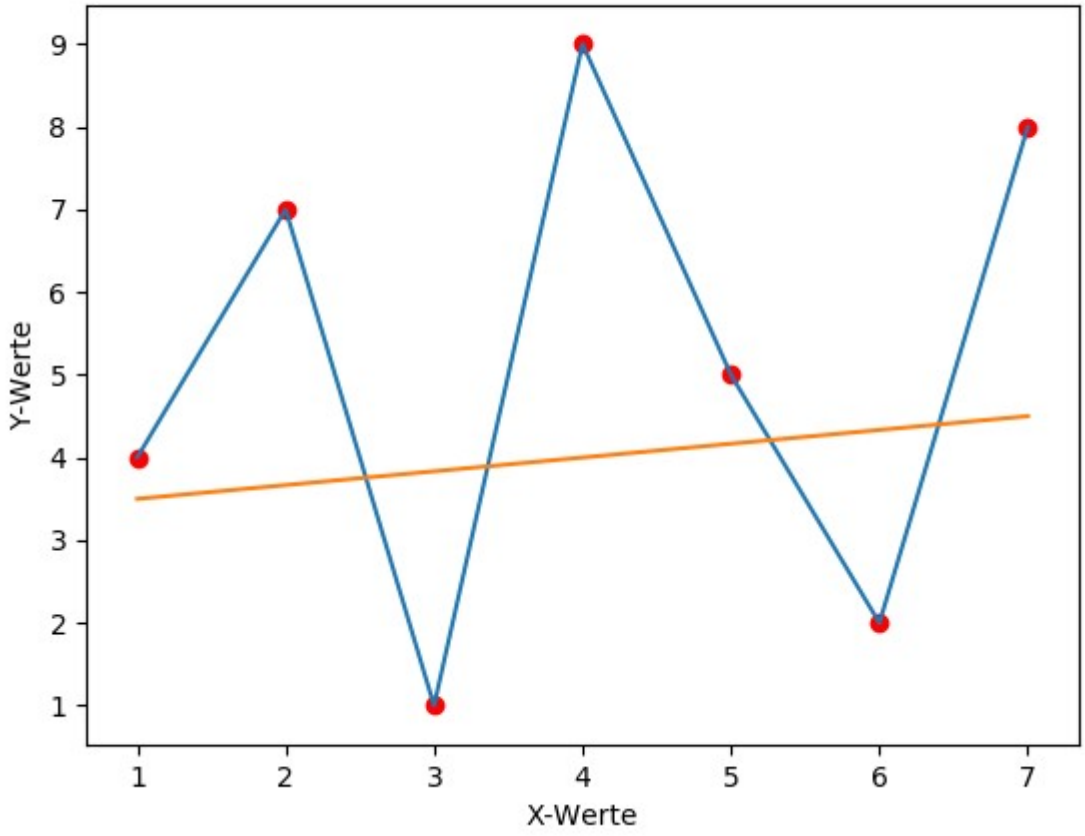
Gerade in Diagramm einzeichnen (z.B. als Trendlinie)

Wollen wir noch eine Gerade in das Diagramm einzeichnen z.B. als Trendlinie, ist dies sehr einfach möglich durch `plt.plot((x1, x2), (y1,y2))`.

```
import matplotlib.pyplot as plt

ywerte = [4, 7, 1, 9, 5, 2, 8]
xwerte = [1, 2, 3, 4, 5, 6, 7]
plt.plot(xwerte, ywerte)
plt.scatter(xwerte, ywerte, color='red')
plt.xlabel("X-Werte")
plt.ylabel("Y-Werte")

x1 = 1
x2 = 7
y1 = 3.5
y2 = 4.5
```

```
plt.plot((x1, x2), (y1, y2))
plt.show()
```

Und somit haben wir automatisch ein drittes Element mit einer automatisch zugewiesenen Farbe:



viele viele Diagrammarten möglich

Die zahlreichen Möglichkeiten sieht man bei den Beispielen unter https://matplotlib.org/gallery - viel Spaß beim Einsatz der Bibliothek matplotlib.