Python für Mathematiker – Matplotlib

Mentoring – WiSe 26/27

Patricia Gerbig
Freie Universität Berlin
Fachbereich Mathematik und Informatik

08. Oktober 2026

Inhalt

1 Matplotlib

2 Plot Design

Inhalt

1 Matplotlib

2 Plot Design

Matplotlib importieren

- Zum Plotten von Daten und Funktionen nutzen wir *Matplotlib*:

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

Source Code 1: Matplotlib importieren

- *Matplotlib* lässt sich mit (je nach Betriebssystem) mit `brew` (MacOS), `pip3` bzw. `winget` (Windows) oder `dnf` (Fedora-Linux) installieren.

Erster Plot

- Mit den Funktionen `plt.plot(...)` und `plt.show()` lässt sich bereits ein erster simpler Plot erstellen.

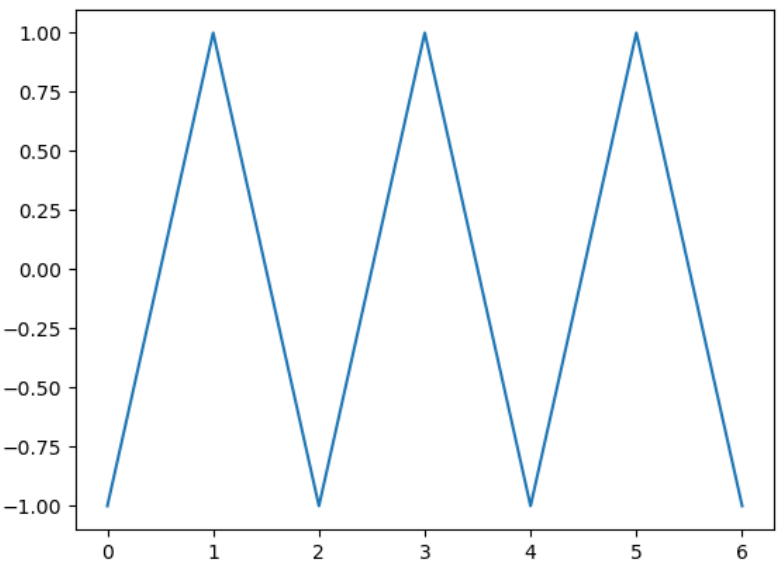

Figure 1: Hütchenfunktion

Code:

```
1  # Indizes dienen hier
       als
2  # Koordinaten f r die
3  # x-Achse
4  plt.plot([-1, 1, -1, 1,
5            -1, 1, -1])
6  # Plot anzeigen
7  plt.show()
```

Source Code 2: Erster Plot

Plot mit Beschriftungen

- Die Funktionen `plt.xlabel(...)` und `plt.ylabel(...)` ermöglichen es die Beschriftungen an den Achsen festzulegen.
- Man kann mit `plt.title(...)` zusätzlich den Titel eines Plots festlegen.

Code:

```
1  # Indizes dienen hier
       als
2  # Koordinaten f r die
3  # x-Achse
4  plt.plot([-1, 1, -1, 1,
5            -1, 1, -1])
6  # x-Achse beschriften
7  plt.xlabel("x-Werte")
8  # y-Achse beschriften
9  plt.ylabel("y-Werte")
10 # Titel festlegen
11 plt.title("Zweiter␣Plot"
       )
12 # Plot anzeigen
13 plt.show()
```

Source Code 3: Beschriftungen festlegen

Plot mit Beschriftungen

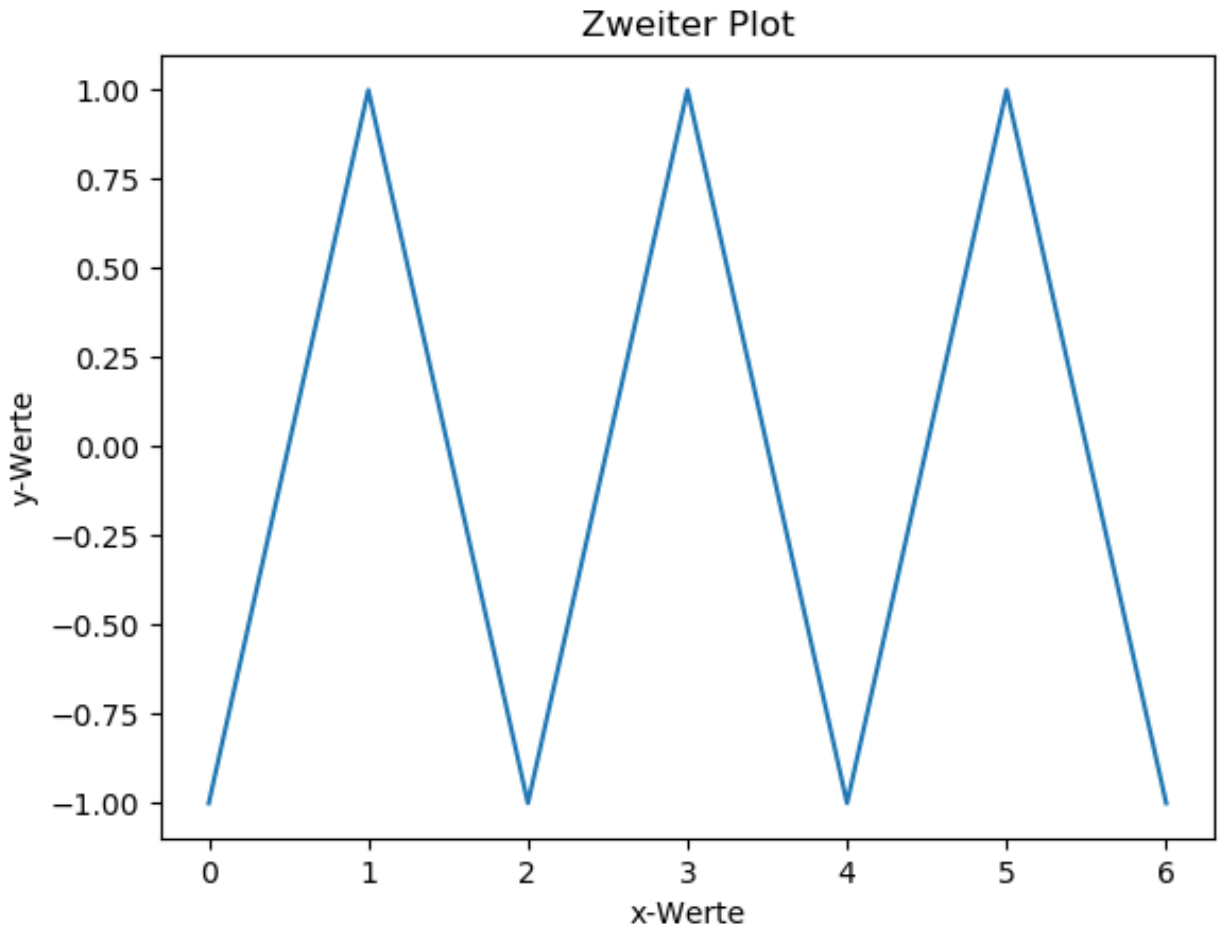

Figure 2: Plot mit Beschriftungen

x-Koor. & Mehrere Graphen

- Die x-Koordinaten lassen sich festlegen, indem man der Funktion `plt.plot(...)` einen zweiten Vektor übergibt. Während der erste Vektor die x-Werte festlegt, gibt der zweite Vektor die y-Werte an,
- Ruft man `plt.plot(...)` mehrfach vor dem Aufruf von `plt.show()` auf, dann erscheinen in einem Plot mehrere Graphen.
- Mit `plt.legend(...)` kann man eine Legende erstellen.

Code:

```
1  # Gitter erstellen
2  x = np.linspace(1, 2)
3  # Mehrere Graphen
       plotten
4  plt.plot(x, x)
5  plt.plot(x, np.exp(x))
6  plt.plot(x, np.log(x)-5)
7  # Legende erstellen
8  plt.legend(["x", "exp(x)
       ", "log(x) - 5"])
9  # Weitere Einstellungen
10 plt.xlabel("x")
11 plt.ylabel("y")
12 plt.title("Plot")
13 # Plot anzeigen
14 plt.show()
```

Source Code 4: Mehrere Graphen

x-Koor. & Mehrere Graphen

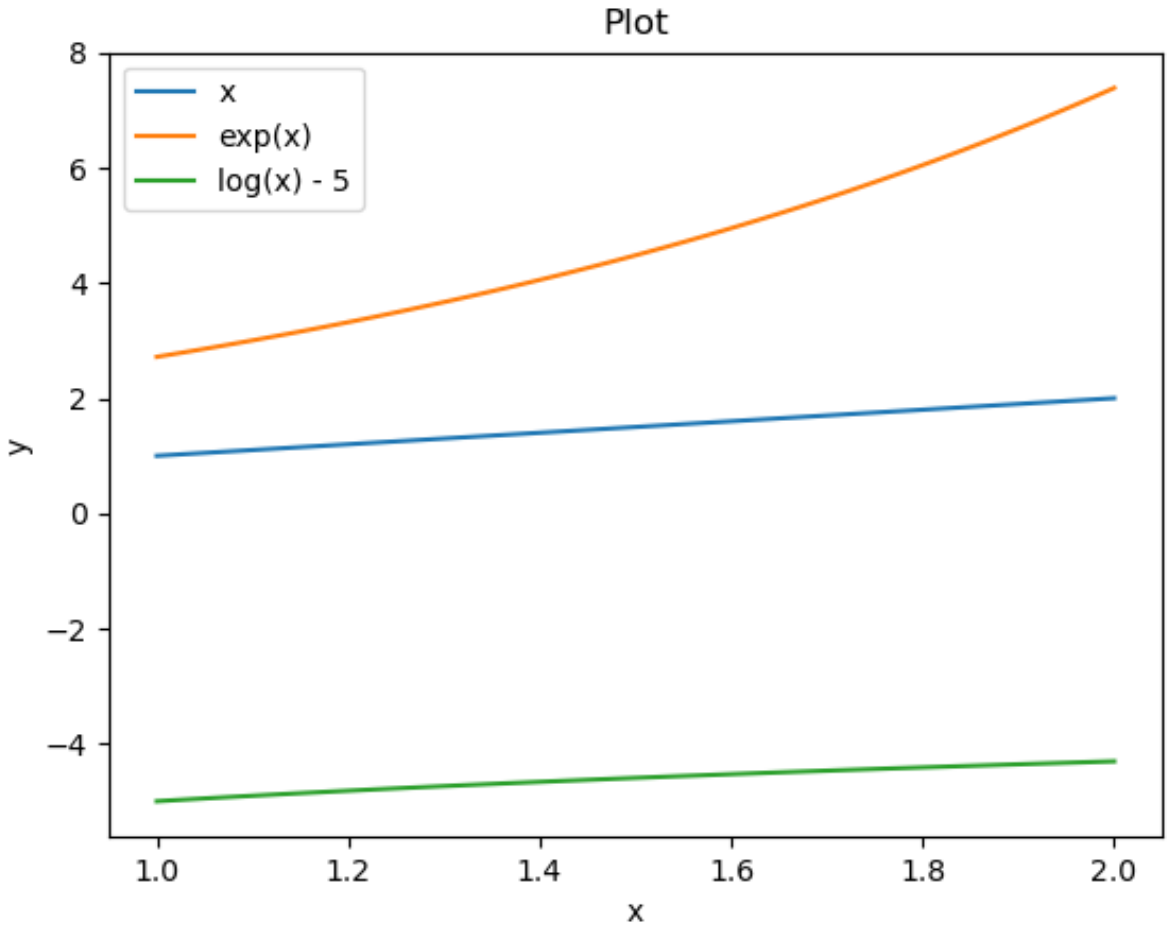

Figure 3: Mehrere Graphen in einem Plot mit Legende

x-Koor. & Mehrere Graphen

- Die Zeilen

```
1  plt.plot(x, x)
2  plt.plot(x, np.exp(x))
3  plt.plot(x, np.log(x)-5)
```

Source Code 5: Mehrere `plt.plot(...)` Aufrufe

sind äquivalent zur Zeile

```
1  plt.plot(x, x, x, np.exp(x), x, np.log(x)-5)
```

Source Code 6: Einziger `plt.plot(...)` Aufruf

Aufgabe 1

Erstelle einen Plot mit den Funktionen x^2 und x^3 in einem Koordinatensystem mit $2 \leq x \leq 5$. Beschrifte die Achsen und den Plot sinnvoll.

Näheres zu `plt.plot(...)`

- In `plt.plot(...)` lässt sich über einen *Format-String* der Form

  `'[marker][line][color]'`

  die Gestalt von einem Graphen festlegen.
- Beispiele:
  - `'.'` – Gepunktete Linie
  - `'g'` – Grüne Linie
  - `'xr'` – Rote Kreuze
  - `'D-y'` – Durchgezogene Linie mit gelben Diamanten
  - `'v--m'` – Magentafarbene gestrichelte Linie mit Dreiecken

Code:

```
1  x = np.linspace(-2, 2,
2                  10)
3  z = np.zeros(10)
4  plt.plot(x, z, ".")
5  plt.plot(x, x-4, "g")
6  plt.plot(x, x**2, "xr")
7  plt.plot(x, x**3, "D-y")
8  plt.plot(x, -3*x, "v--m"
       )
9  plt.legend(["0", "$x-4$"
       , "$x^2$", "$x^3$",
       "$-3\cdot␣x$"])
10 plt.show()
```

Source Code 7: Verschiedene Format-Strings

Inhalt

1 Matplotlib

2 Plot Design

Näheres zu `plt.plot()`

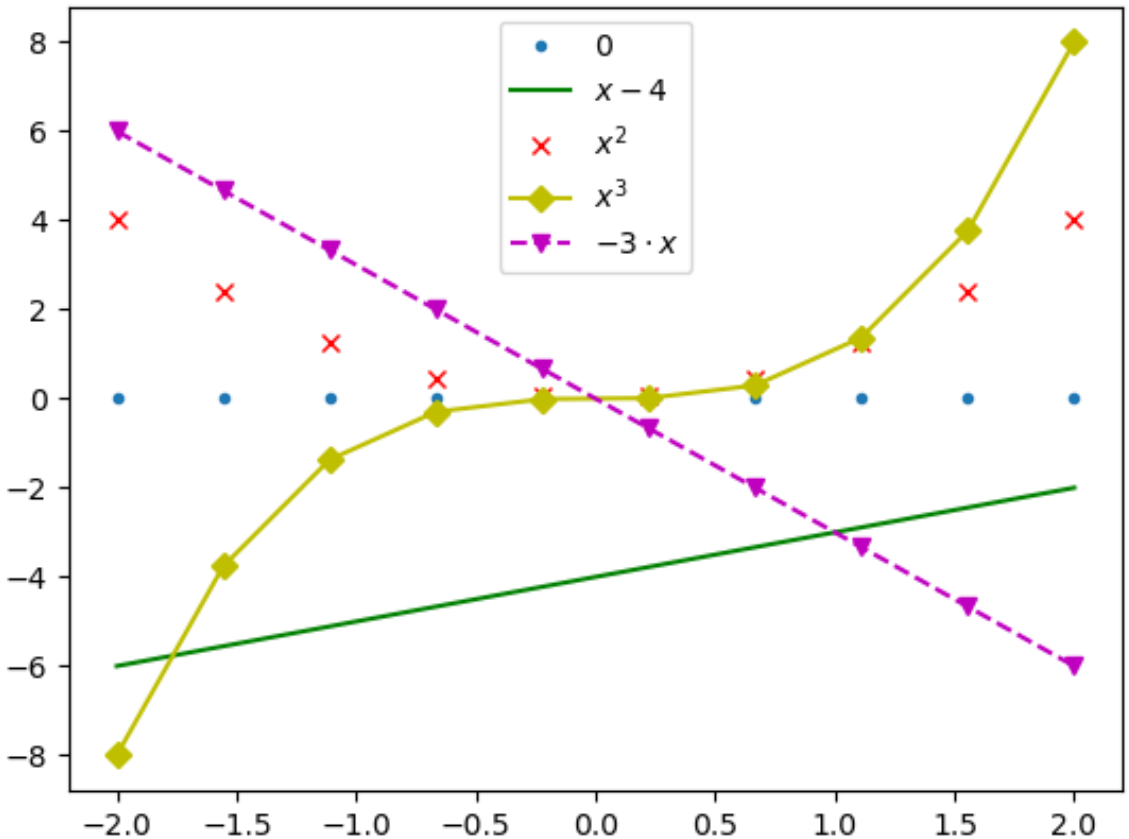

Figure 4: Farbiger Plot

Näheres zu `plt.plot()`

`fmt = '[marker][line][color]'`

marker	
`'.'`	Punkt für jeden Gitterpunkt
`'o'`	Kreis für jeden Gitterpunkt
`'s'`	Quadrat für jeden Gitterpunkt
`'*'`	Stern für jeden Gitterpunkt
`'x'`	Kreuz für jeden Gitterpunkt
`'D'`	Diamand für jeden Gitterpunkt
line	
`-`	Durchgezogene Linie
`--`	Gestrichelte Linie
`-.`	Strichpunktierte Linie
`:`	Gepunktete Linie

Table 1: Optionen für Format-String

Näheres zu `plt.plot()`

`fmt = '[marker][line][color]'`

color	
`'b'`	Blau
`'g'`	Grün
`'r'`	Rot
`'c'`	Cyan
`'m'`	Magenta
`'y'`	Gelb
`'k'`	Schwarz
`'w'`	Weiß

Table 2: Optionen für Format-String

Aufgabe 2

Ändere in deinem Plot aus Aufgabe 1 (x^2 und x^3 mit $2 \leq x \leq 5$) die Farben. x^3 soll eine gestrichelte Linie werden.

Weitere Plot-Funktionen

- Zur besseren Darstellung eines Graphen bietet es sich in manchen Situationen an die x- oder y-Achse *logarithmisch* zu skalieren.
- Den Funktionen in der Tabelle kann ein *Format-String* übergeben werden.

color	
`plt.semilogx(...)`	Plot mit logarithmisch skalierter x-Achse
`plt.semilogy(...)`	Plot mit logarithmisch skalierter y-Achse
`plt.loglog(...)`	Plot mit logarithmisch skalierter x- und y-Achse

Table 3: Nahe Verwandten von `plt.plot(...)`

- Beispiel:

```
1  x = np.arange(20)
2  y = np.array([np.math.factorial(n) for n in x])
3  plt.semilogy(x, y, "o-r")
4  plt.show()
```

Source Code 11: Beispiel für `plt.semilogy(...)`

Bsp. zu `plt.semilogy(...)`

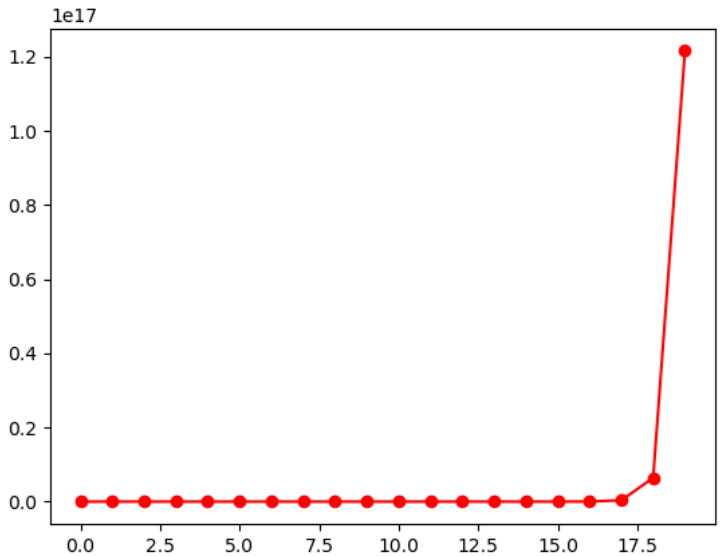

Figure 6: Fakultätsfunktion mit `plt.plot(x, y)`

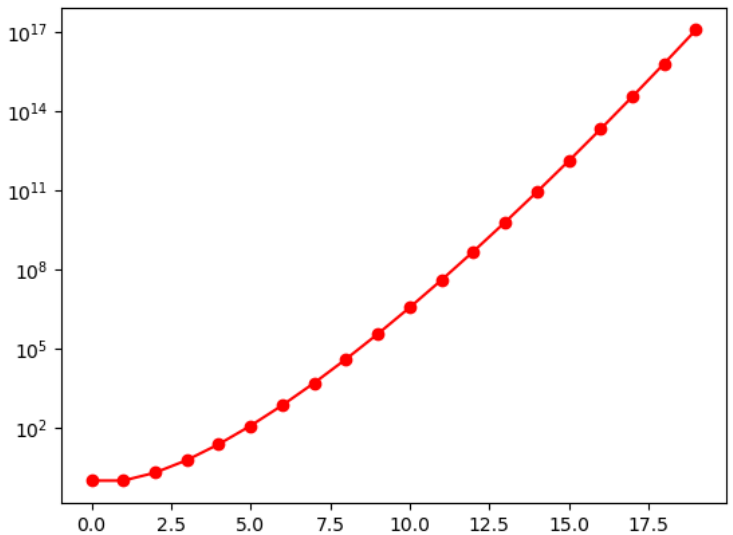

Figure 7: Fakultätsfunktion mit `plt.semilogy(x, y)`

Gitter

- In einem Plot lässt sich mit `plt.grid(...)` ein Gitter anzeigen.

Code:

```
1  x = np.linspace(-50, 50)
2  plt.grid()
3  plt.plot(x, x)
4  plt.show()
```

Source Code 12: Gitter anzeigen

Gitter

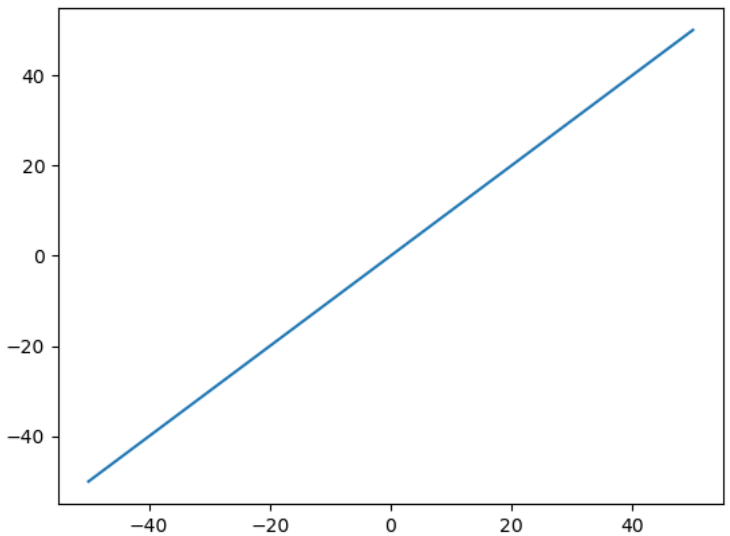

Figure 8: Ohne `plt.grid()`

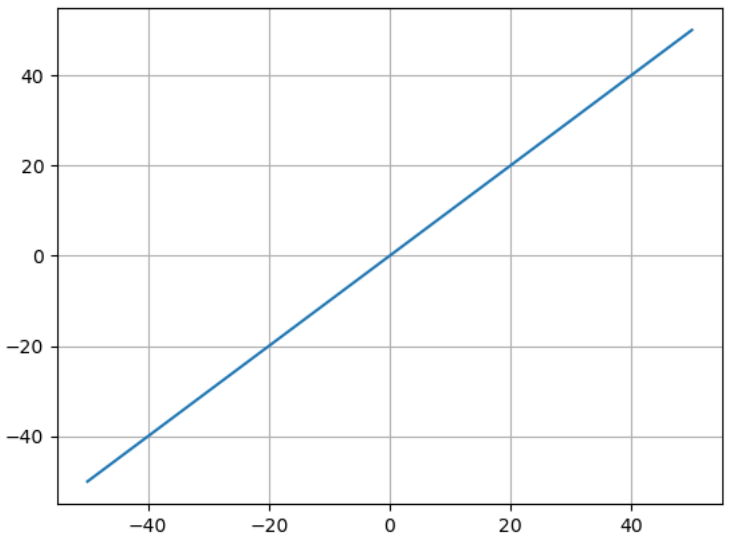

Figure 9: Mit `plt.grid()`

Skalen festlegen

- Die Achsenskalierungen lassen sich mit der Funktion

  `plt.axis(...)`

  festlegen.

Code:

```
1  x = np.linspace(-1, 1)
2  y = np.exp(x)
3  # x-Achse geht von
4  # -2 bis 2 y-Achse
5  # geht von 0 bis 3
6  xmin, xmax = -2, 2
7  ymin, ymax = 0, 3
8  plt.axis([xmin, xmax,
9            ymin, ymax])
10 plt.plot(x, y)
```

Source Code 13: Achsenskallierung

Skalen festlegen

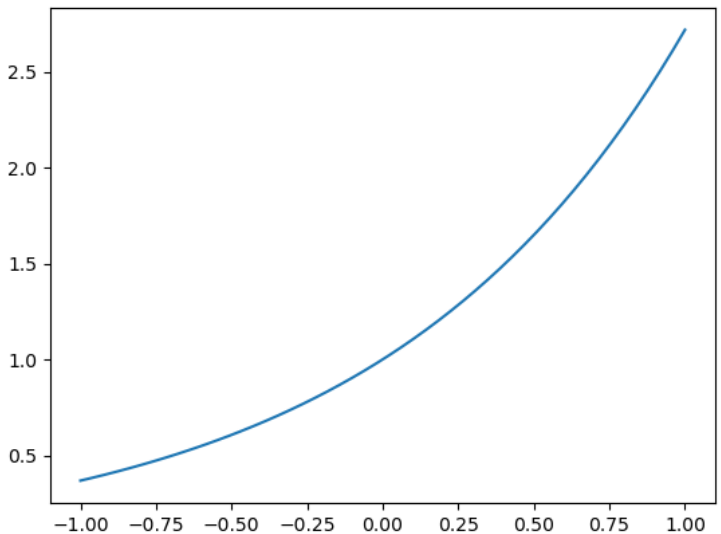

Figure 10: Ohne `plt.axis([-2, 2, 0, 3])`

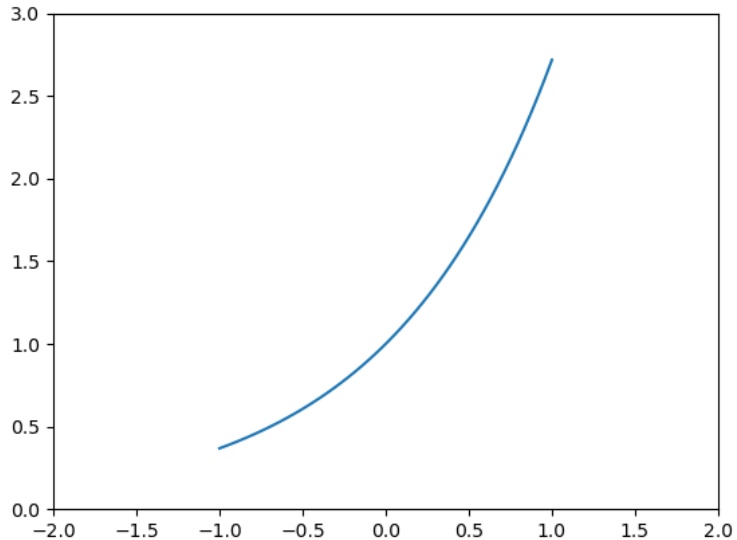

Figure 11: Mit `plt.axis([-2, 2, 0, 3])`

Subplots

- Für mehrere Subplots in einem Plot benutzt man `plt.subplot(xyz)`.
- Die Subplots werden über eine Ziffernfolge *xyz* gesteuert, wobei
  - $\mathtt{x}$ = Anzahl von Spalten,
  - $\mathtt{y}$ = Anzahl von Zeilen und
  - $\mathtt{z}$ = Nummer des Subplots

  meint.
- Beispiel: Möchte man ein (2×3)-Schema von Plots, muss $\mathtt{x} = 2$ und $\mathtt{y} = 3$ sein. Möchte man den fünften Subplot ansprechen, muss man $\mathtt{z} = 5$ setzen. Mit dem Befehl `plt.subplot(235)` kann man den fünften Subplot im (2×3)-Schema bearbeiten.

Subplots


```
1  x = np.linspace(0, 1, 200)
2  plt.subplot(121)
3  plt.plot(x, x**2, "r", x, np.sqrt(x), "g")
4  plt.title("Parabel und Wurzel")
5  plt.xlabel("x")
6  plt.ylabel("Wert")
7  plt.legend(["$x^2$", "sqrt(x)"])
8  plt.grid()
9  plt.subplot(122)
10 plt.plot(x, np.exp(x), "-.m")
11 plt.title("Exponentialfunktion")
12 plt.xlabel("x")
13 plt.ylabel("Wert")
14 plt.legend(["exp(x)"])
15 plt.grid()
16 plt.show()
```

Source Code 14: Beispiel für `plt.subplot(...)`

Subplots

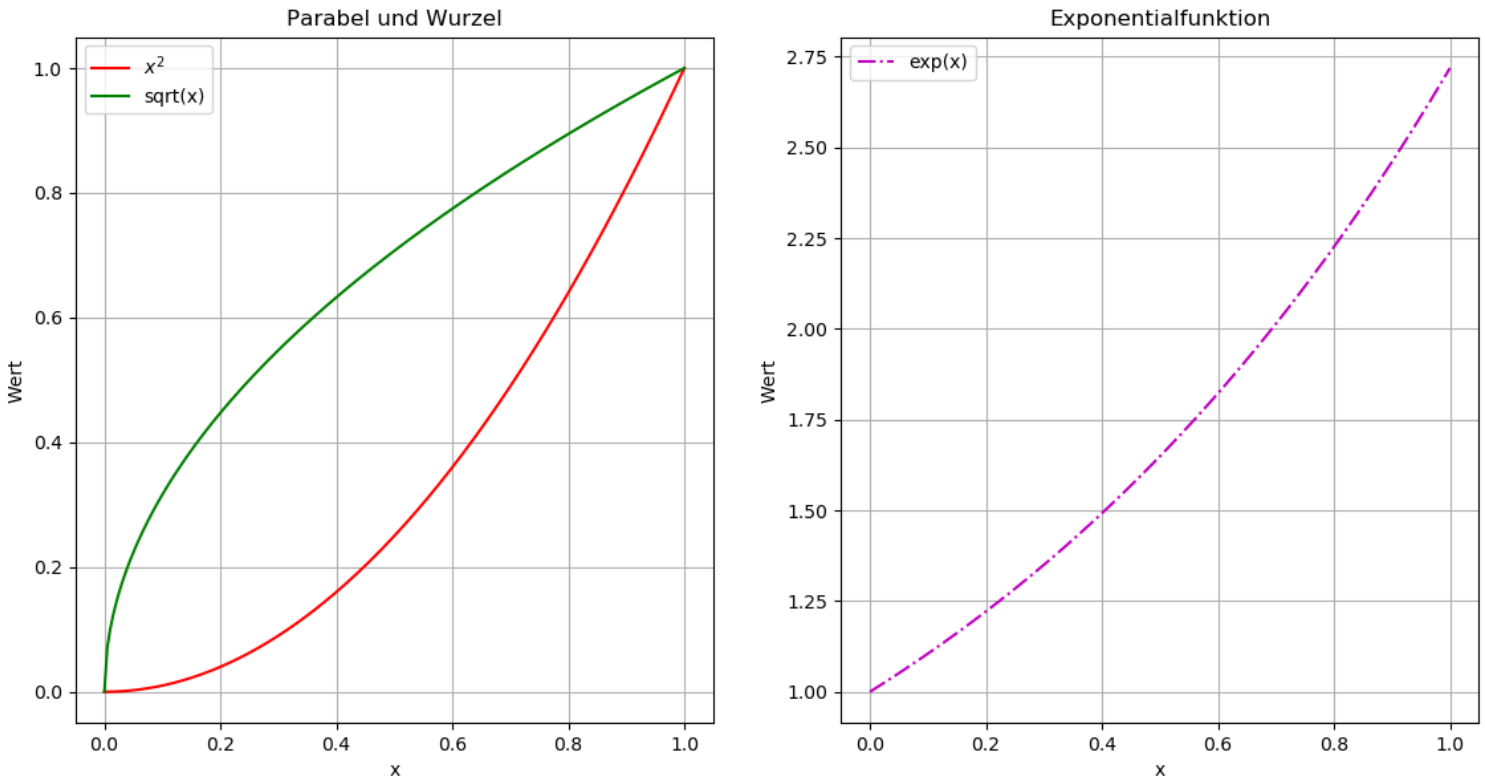

Figure 12: Subplots

Aufgabe 3

Extrahiere deine Graphen aus Aufgabe 2 (x^2 und x^3 mit $2 \leq x \leq 5$) in zwei nebeneinanderstehende Subplots.

Aufgabe 3

1. 'Male' mind. ein Symbol in einen Plot. (z.B. ein Stern, Herz oder das Haus vom Nikolaus)
   Überlege dir zuerst, welche Teile du durch welche Funktion oder Datenpunkte darstellen kannst.
2. Extraaufgabe: Verschönere deinen Plot, in dem die Symbole z.B. ausgemahlt werden (guck dir dafür z.B. plt.fill() an).

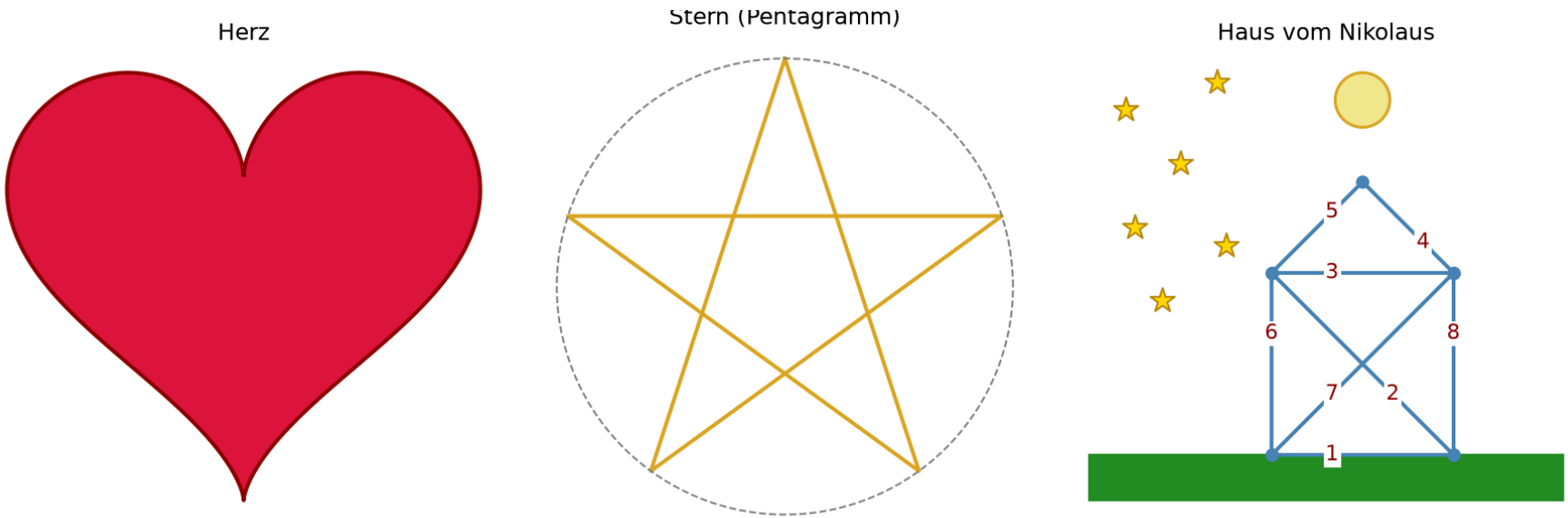

Figure 14: Subplots