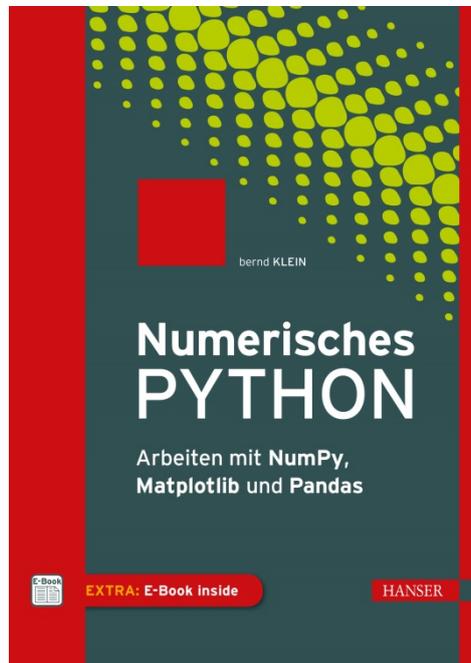


HANSER



Leseprobe

zu

„Numerisches Python“

von Bernd Klein

Print-ISBN: 978-3-446-45076-9
E-Book-ISBN: 978-3-446-45363-0

Weitere Informationen und Bestellungen unter
<http://www.hanser-fachbuch.de/978-3-446-45076-9>

sowie im Buchhandel

© Carl Hanser Verlag, München

Inhalt

Vorwort	XV
Danksagung	XVI
1 Einleitung	1
1.1 Die richtige Wahl	1
1.2 Aufbau des Buches	2
1.3 Python-Installation	3
1.4 Download der Beispiele	3
1.5 Anregungen und Kritik	3
2 Numerisches Programmieren mit Python	5
2.1 Definition von numerischer Programmierung	5
2.2 Zusammenhang zwischen Python, NumPy, Matplotlib, SciPy und Pandas	6
2.3 Python, eine Alternative zu Matlab	7
Teil I Kurze Einführung in Python	9
3 Kurze Einführung in Python	11
3.1 Datenstrukturen	11
3.1.1 Zahlen und Variablen	11
3.1.2 Zeichenketten	12
3.1.3 Listen	15
3.1.4 Tupel	16
3.1.5 Frozensets und Mengen in Python	17
3.1.6 Dictionaries	18
3.2 Kontrollstrukturen	19
3.2.1 Bedingte Anweisungen	19
3.2.2 Schleifen	20

3.2.3	Funktionen	23
3.3	Ausnahmebehandlung	26
3.3.1	Die optionale else-Klausel	29
3.3.2	Exceptions generieren	29
3.3.3	Finalisierungsaktion	29
3.4	Dateien lesen und schreiben	30
3.4.1	Datei lesen.....	30
3.4.2	Datei schreiben	31
3.5	Modularisierung	32
3.5.1	Namensräume von Modulen	32
3.5.2	Suchpfad für Module	33
3.5.3	Inhalt eines Moduls	33
3.5.4	Eigene Module	33
3.5.5	Dokumentation für eigene Module	34
3.6	Klassen-Definition	35
3.6.1	Eine einfache Klasse	35
3.6.2	Attribute	35
3.6.3	Initialisierung von Instanzen	37
3.6.4	Vererbung	37
3.6.5	Private, geschützte und öffentliche Attribute	38
3.6.6	Properties.....	39
Teil II NumPy		41
4	NumPy Einführung	43
4.1	Überblick.....	43
4.2	Vergleich NumPy-Datenstrukturen und Python.....	44
4.3	Ein einfaches Beispiel	44
4.4	Grafische Darstellung der Werte	45
4.5	Speicherbedarf	46
4.6	Zeitvergleich zwischen Python-Listen und NumPy-Arrays	49
5	Arrays in NumPy erzeugen.....	51
5.1	Erzeugung äquidistanter Intervalle	51
5.1.1	arange	51
5.1.2	linspace	52
5.1.3	Nulldimensionale Arrays in NumPy	53

5.1.4	Eindimensionales Array	53
5.1.5	Zwei- und Mehrdimensionale Arrays.....	54
5.2	Shape/Gestalt eines Arrays	54
5.3	Indizierung und Teilbereichsoperator	56
5.4	Dreidimensionale Arrays	61
5.5	Arrays mit Nullen und Einsen	64
5.6	Arrays kopieren.....	65
5.6.1	numpy.copy(A) und A.copy()	65
5.6.2	Zusammenhängend gespeicherte Arrays	65
5.7	Identitätsarray	67
5.7.1	Die identity-Funktion.....	67
5.7.2	Die eye-Funktion	68
5.8	Aufgaben	69
5.9	Lösungen.....	70
6	Datentyp-Objekt: dtype	73
6.1	dtype.....	73
6.2	Strukturierte Arrays.....	75
6.3	Ein- und Ausgabe von strukturierten Arrays	77
6.4	Unicode-Strings in Arrays	79
6.5	Umbenennen von Spaltennamen	79
6.6	Spaltenwerte austauschen.....	80
6.7	Komplexeres Beispiel.....	80
6.8	Aufgaben	82
6.9	Lösungen.....	83
7	Dimensionsänderungen.....	85
7.1	Reduktion und Reshape von Arrays	85
7.1.1	flatten	86
7.1.2	ravel	86
7.1.3	Unterschiede zwischen ravel und flatten	87
7.1.4	reshape	88
7.2	Konkatenation von Arrays	89
7.3	Weitere Dimensionen hinzufügen.....	90
7.4	Vektoren stapeln	91
7.5	„Fliesen“ mit „tile“	92

8	Numerische Operationen auf NumPy-Arrays	95
8.1	Operatoren und Skalare	95
8.2	Arithmetische Operationen auf zwei Arrays	97
8.3	Matrizenmultiplikation und Skalarprodukt	98
8.3.1	Definition der dot-Funktion	98
8.3.2	Beispiele zur dot-Funktion	98
8.3.3	Das dot-Produkt im 3-dimensionalen Fall	100
8.4	Vergleichsoperatoren	105
8.5	Logische Operatoren	106
8.6	Broadcasting	106
8.6.1	Zeilenweises Broadcasting	107
8.6.2	Spaltenweises Broadcasting	109
8.6.3	Broadcasting von zwei eindimensionalen Arrays	110
8.7	Distanzmatrix	111
8.8	Ufuncs	112
8.8.1	Anwendung von Ufuncs	112
8.8.2	Parameter für Rückgabewerte bei Ufuncs	114
8.8.3	accumulate	115
8.8.4	reduce	116
8.8.5	outer	117
8.8.6	at	118
8.9	Aufgaben	118
8.10	Lösungen	119
9	Statistik und Wahrscheinlichkeiten	121
9.1	Einführung	121
9.2	Zufallszahlen mit Python	122
9.2.1	Die Module random und secrets	122
9.2.2	Erzeugen einer Liste von Zufallszahlen	122
9.3	Zufällige Integer-Zahlen mit Python	123
9.4	Stichproben/Auswahlen	126
9.5	Zufallsintervalle	127
9.6	Gewichtete Zufallsauswahl	128
9.7	Stichproben mit Python	130
9.8	Kartesische Auswahl	132
9.8.1	Kartesisches Produkt	132
9.8.2	Kartesische Auswahl: cartesian_choice	133
9.9	Echte Zufallszahlen	135

9.10	Seed/Startwert	136
9.11	Gauss'sche Normalverteilung	137
9.12	Übung mit Binärsender	139
9.13	Synthetische Verkaufszahlen	141
9.14	Aufgaben	143
9.15	Lösungen	145
10	Boolesche Maskierung und Indizierung	151
10.1	Fancy-Indizierung	153
10.2	Indizierung mit einem Integer-Array	153
10.2.1	Übung	154
10.2.2	Lösung	154
10.3	nonzero und where	154
10.3.1	Übung	155
10.3.2	Lösung	156
10.3.3	Flatnonzero und count_nonzero	156
11	Lesen und Schreiben von Datendateien	157
11.1	Textdateien speichern mit savetxt	157
11.2	Textdateien laden mit loadtxt	159
11.2.1	loadtxt ohne Parameter	159
11.2.2	Spezielle Trenner	159
11.2.3	Selektives Einlesen von Spalten	159
11.2.4	Datenkonvertierung beim Einlesen	160
11.3	tofile	161
11.4	fromfile	162
11.5	Best Practice, um Daten zu laden und zu speichern	164
11.6	Und noch ein anderer Weg: genfromtxt	164
Teil III	Matplotlib	165
12	Einführung	167
12.1	Ein erstes Beispiel	168
12.2	Der Formatparameter von pyplot.plot	169
12.3	Bezeichnungen für die Achsen	172
12.4	Abfragen und Ändern des Wertebereichs der Achsen	174
12.5	„linspace“ zur Definition von X-Werten	175
12.6	Linienstil ändern	177
12.7	Flächen einfärben	178

13	Achsen- und Skalenteilung	181
13.1	Achsenverschiebungen und Achsenbezeichnungen.....	181
13.2	Verändern der Achsenbeschriftungen	186
13.3	Justierung der Tick-Beschriftungen	187
14	Legenden und Kommentare hinzufügen	189
14.1	Legende hinzufügen.....	189
14.2	Kommentare	194
15	Mehrfache Plots und Doppelachsen	199
15.1	Mehrere Abbildungen und Achsen	199
15.2	Unterdigramm mit gridspec	208
15.3	Arbeiten mit Objekten.....	211
15.4	Ein Plot innerhalb eines anderen Plots	213
15.5	Setzen des Plotbereichs	214
15.6	Logarithmische Darstellung.....	215
15.7	Sekundäre Y-Achse	216
15.8	Gitterlinien.....	217
15.9	Abbildungen speichern	218
15.10	Aufgaben	219
15.11	Lösungen.....	219
16	Konturplots	223
16.1	Erstellen eines Maschengitters.....	223
16.2	Berechnung der Werte	225
16.3	Linienstil und Farben anpassen	227
16.4	Gefüllte Konturen	228
16.5	Individuelle Farben	229
16.6	Schwellen	230
16.7	Andere Grids.....	231
16.7.1	Meshgrid genauer	231
16.7.2	mgrid.....	235
16.7.3	ogrid.....	235
16.8	Aufgaben	236
16.9	Lösungen.....	237
17	Balken-, Säulendiagramme und Histogramme	241
17.1	Histogramme	241
17.2	Säulendiagramm.....	247

17.3	Balkendiagramme	249
17.4	Aufgaben	250
17.5	Lösung.....	250

Teil IV Pandas251

18 Einführung in Pandas 253

18.1	Datenstrukturen	253
18.2	Series	254
18.2.1	Indizierung.....	256
18.2.2	pandas.Series.apply	257
18.2.3	Zusammenhang zu Dictionaries	258
18.3	NaN – Fehlende Daten	258
18.3.1	Die Methoden isnull() und notnull()	259
18.3.2	Zusammenhang zwischen NaN und None	260
18.3.3	Fehlende Daten filtern.....	260
18.3.4	Fehlende Daten auffüllen	261

19 DataFrame 263

19.1	Zusammenhang zu Series	263
19.2	Manipulation der Spaltennamen	264
19.3	Zugriff auf Spalten	265
19.4	DataFrames aus Dictionaries	266
19.5	Index ändern	267
19.5.1	Umsortierung der Spalten	267
19.5.2	Spalte in Index umfunktionieren.....	269
19.6	Selektion von Zeilen	270
19.7	Summen und kumulative Summen	271
19.8	Spaltenwerte ersetzen	271
19.9	Sortierung.....	273
19.10	Spalten einfügen	274
19.11	DataFrame und verschachtelte Dictionaries.....	275
19.12	Aufgaben	276
19.13	Lösungen.....	278

20 Dateien lesen und schreiben 283

20.1	Trennerseparierte Werte	283
20.2	CSV- und DSV-Dateien lesen	284
20.3	Schreiben von CSV-Dateien	285

20.4	Lesen und Schreiben von Excel-Dateien	287
20.5	Aufgaben	288
20.6	Lösungen.....	288
21	Umgang mit NaN	291
21.1	'nan' in Python	291
21.2	NaN in Pandas.....	292
21.2.1	Beispiel mit NaNs	294
21.3	dropna() verwenden.....	295
21.4	Aufgaben	297
21.5	Lösungen.....	297
22	Binning	299
22.1	Einführung	299
22.2	Binning mit Pandas	301
22.2.1	Von Pandas verwendete Bins	301
22.2.2	Andere Wege, um Bins zu definieren	302
22.2.3	Bins und Werte zählen.....	303
22.2.4	Bins benennen.....	305
23	Mehrstufige Indizierung	307
23.1	Einführung	307
23.2	Mehrstufig indizierte Series	307
23.3	Zugriffsmöglichkeiten	309
23.4	Zusammenhang zu DataFrames	310
23.5	Dreistufige Indizes	312
23.6	Vertauschen mehrstufiger Indizes.....	314
23.7	Aufgaben	315
23.8	Lösungen.....	316
24	Datenvisualisierung mit Pandas	317
24.1	Einführung	317
24.2	Liniendiagramme in Pandas	318
24.2.1	Series	318
24.2.2	DataFrames	320
24.2.3	Sekundärachsen (Twin Axes)	322
24.2.4	Mehrere Y-Achsen	323
24.3	Ein komplexeres Beispiel	325
24.3.1	Spalten mit Zeichenketten (Strings) in Floats wandeln	328

24.4	Balkendiagramme in Pandas	329
24.4.1	Ein einfaches Beispiel.....	329
24.4.2	Balkengrafik für die Programmiersprachennutzung	330
24.4.3	Farbgebung einer Balkengrafik	331
24.5	Kuchendiagramme in Pandas	332
24.5.1	Ein einfaches Beispiel.....	332
25	Zeit und Datum	337
25.1	Einführung.....	337
25.2	Python-Standardmodule für Zeitdaten	338
25.2.1	Die date-Klasse	338
25.2.2	Die time-Klasse	339
25.3	Die datetime-Klasse	340
25.4	Unterschied zwischen Zeiten	342
25.4.1	Wandlung von datetime-Objekten in Strings.....	343
25.4.2	Wandlung mit strptime.....	343
25.5	Ausgabe in Landessprache	344
25.6	datetime-Objekte aus Strings erstellen	345
26	Zeitserien	347
26.1	Einführung.....	347
26.2	Zeitreihen und Python	347
26.3	Datumsbereiche erstellen	350
	Stichwortverzeichnis	353

Vorwort

Eine der treibenden Kräfte in der weltweiten Softwareentwicklung wird wohl am besten durch die beiden populären Begriffe „Big Data“ und „Maschinelles Lernen“ beschrieben. Immer mehr Institute und Firmen betätigen sich in diesen Feldern. Für diese und auch für individuelle Personen, die in diesen Bereichen tätig werden wollen, ist eine der bedeutendsten Fragen – wenn nicht gar die bedeutendste Frage –, was die geeignetste Programmiersprache zu diesem Zweck ist. In vielen Umfragen wird Python als beste oder auch als beliebteste Programmiersprache genannt.

Python war ursprünglich nicht für numerische Probleme ausgerichtet gewesen. Die Erfolgsstory von Python wurde erst möglich durch die Module NumPy, SciPy, Matplotlib und Pandas. Dieses Buch bietet eine umfassende Einführung in die Module NumPy, Matplotlib und Pandas, setzt aber grundlegende Kenntnisse von Python voraus. Somit ergänzt es in idealer Weise das Buch „Einführung in Python 3: Für Ein- und Umsteiger“ von Bernd Klein.

Brigitte Bauer-Schiewek, Lektorin

Matplotlib ist eine Bibliothek zum Plotten wie GNUplot. Der Hauptvorteil gegenüber GNUplot ist die Tatsache, dass es sich bei Matplotlib um ein Python-Modul handelt. Aufgrund des wachsenden Interesses an der Programmiersprache Python steigt auch die Popularität von Matplotlib kontinuierlich.

Ein anderer Grund für die Attraktivität von Matplotlib liegt in der Tatsache, dass es als gute Alternative, wenn es ums Plotten geht, für MATLAB angesehen wird, wenn es in Verbindung mit NumPy und SciPy benutzt wird. Während es sich bei MATLAB um kostspielige Closed-Source-Software handelt, ist die Software von Matplotlib frei, kostenlos und quelloffen. Außerdem kann in Matplotlib objektorientiert programmiert werden. Es kann auch in allgemeinen GUIs wie wxPython, Qt und GTK+ verwendet werden. Mit der „pylab“-Erweiterung wird die Möglichkeit geboten, noch MATLAB-ähnlicher zu programmieren. Davon wird jedoch im Allgemeinen abgeraten, da dies zu einem unsauberem Programmierstil führt, auch wenn es dadurch MATLAB-Nutzern extrem leicht gemacht wird zu wechseln.

Mittels Matplotlib kann man Diagramme und Darstellungen in verschiedenen Formaten erzeugen, die dann in Veröffentlichungen verwendet werden können.

Eine andere Besonderheit besteht in der steilen Lernkurve, was sich darin zeigt, dass die Benutzerinnen und Benutzer sehr schnelle Fortschritte bei der Einarbeitung machen. Auf der offiziellen Webseite steht dazu Folgendes: „Matplotlib versucht, Einfaches einfach und Schweres möglich zu machen. Man kann mit nur wenigen Codezeilen Plots, Histogramme,

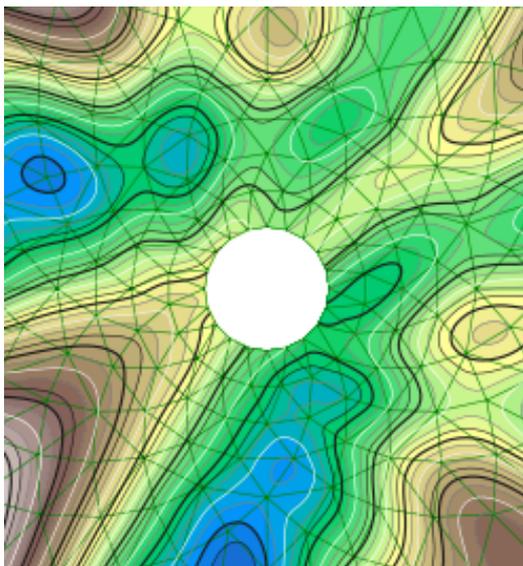


Bild 12.1 Tricontouring

Leistungsspektren, Balkendiagramme, Fehlerdiagramme, Streudiagramme (Punktewolken) und so weiter erzeugen.“¹

■ 12.1 Ein erstes Beispiel

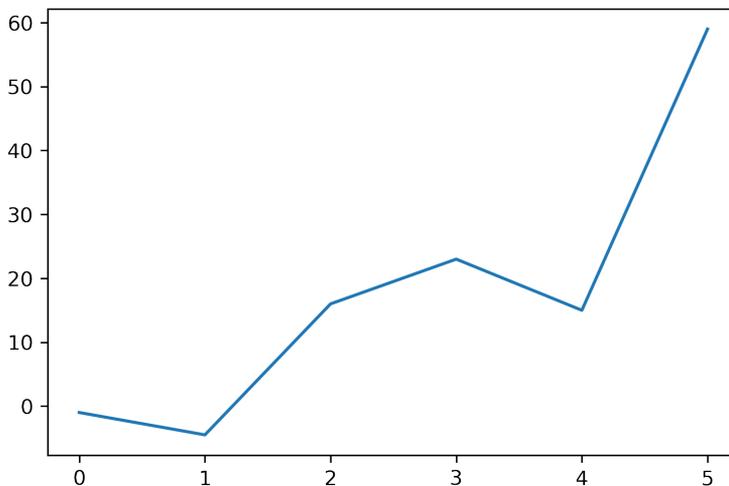
Wir werden mit einem einfachen Graphen beginnen. So einfach, dass es nicht mehr einfacher geht. Ein Graph in Matplotlib ist eine zwei- oder dreidimensionale Zeichnung, die mithilfe von Punkten, Kurven, Balken oder anderem einen Zusammenhang herstellt. Es gibt zwei Achsen: die horizontale x-Achse für die unabhängigen Werte und die vertikale y-Achse für die abhängigen Werte.

Wir werden im Folgenden das Untermodul `pyplot` verwenden. `pyplot` stellt eine prozedurale Schnittstelle zur objektorientierten Plot-Bibliothek von Matplotlib zur Verfügung. Die Kommandos von `pyplot` sind so gewählt, dass sie sowohl in den Namen als auch in ihren Argumenten MATLAB ähnlich sind.

Es ist allgemein üblich, `matplotlib.pyplot` in `plt` umzubenennen. In unserem ersten Beispiel werden wir die `plot`-Funktion von `pyplot` benutzen. Wir übergeben an die `plot`-Funktion eine Liste von Werten. `plot` betrachtet und benutzt die Werte dieser Liste als y-Werte. Die Indizes dieser Liste werden automatisch als x-Werte genommen.

```
import matplotlib.pyplot as plt

plt.plot([-1, -4.5, 16, 23, 15, 59])
plt.show()
```



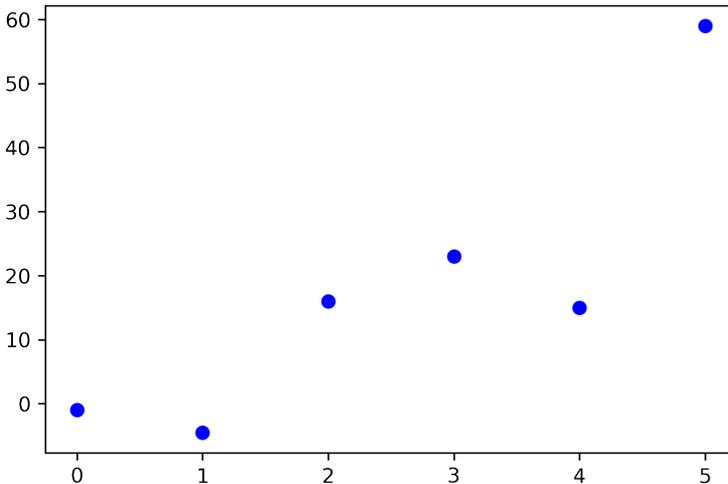
¹ „Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.“

Wir sehen einen zusammenhängenden Graphen, obwohl wir nur diskrete Werte für die Ordinate, allgemein auch als Y-Achse bezeichnet, zur Verfügung gestellt hatten. Als Werte für die Abszisse, also die X-Achse, wurden die Indizes genommen.

Indem wir einen Formatstring beim Funktionsaufruf mit übergeben, können wir einen Graphen mit diskreten Werten erzeugen, in unserem Fall mit blauen Vollkreisen. Der Formatstring definiert, wie die diskreten Punkte darzustellen sind:

```
import matplotlib.pyplot as plt

plt.plot([-1, -4.5, 16, 23, 15, 59], "ob")
plt.show()
```



■ 12.2 Der Formatparameter von pyplot.plot

In unserem vorigen Beispiel hatten wir `ob` als Formatparameter genutzt. Er besteht aus zwei Zeichen. Das erste definiert den Linienstil oder die Darstellung der diskreten Werte, die Markierungen (englisch „markers“). Mit dem zweiten Zeichen wählt man die Farbe für den Graphen aus. Die Reihenfolge der beiden Zeichen hätte aber auch umgekehrt sein können, d.h. wir hätten auch `bo` schreiben können. Falls kein Formatparameter angegeben wird, wie es in unserem ersten Beispiel der Fall war, wird `b-` als Default-Wert benutzt, d.h. eine durchgehende blaue Linie wird ausgegeben.

Die folgenden Zeichen werden in einem Formatstring akzeptiert, um den Linienstil oder die Marker zu steuern:

Zeichen	Beschreibung
'-' (Bindestrich)	durchgezogene Linie
'_' (zwei Bindestriche)	gestrichelte Linie
'-.'	Strichpunkt-Linie
':'	punktierte Linie
':'	Punkt-Marker
','	Pixel-Marker
'o'	Kreis-Marker
'v'	Dreiecks-Marker, Spitze nach unten
'^'	Dreiecks-Marker, Spitze nach oben
'<'	Dreiecks-Marker, Spitze nach links
'>'	Dreiecks-Marker, Spitze nach rechts
'1'	tri-runter-Marker
'2'	tri-hoch-Marker
'3'	tri-links Marker
'4'	tri-rechts Marker
's'	quadratischer Marker
'p'	fünfeckiger Marker
'*'	Stern-Marker
'h'	Sechseck-Marker1
'H'	Sechseck-Marker2
'+'	Plus-Marker
'x'	x-Marker
'D'	rautenförmiger Marker
'd'	dünnere rautenförmiger Marker
' '	Marker in Form einer vertikalen Linie
'_'	Marker in Form einer horizontalen Linie

Die folgenden Farbabkürzungen sind möglich:

Zeichen	Farbe
'b'	blau
'g'	grün
'r'	rot
'c'	cyan
'm'	magenta
'y'	gelb
'k'	schwarz
'w'	weiß

Wie einige sicherlich schon vermutet haben, kann man auch X-Werte an die Plot-Funktion übergeben. Im folgenden Beispiel übergeben wir eine Liste mit den Vielfachen von 3 zwischen 0 und 21 als X-Werte an plot:

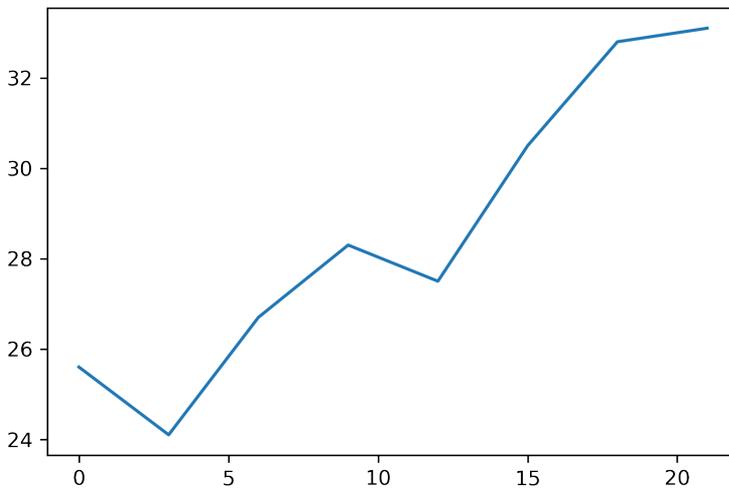
```
import matplotlib.pyplot as plt

# die X-Werte:
days = list(range(0, 22, 3))
print(days)
# die Y-Werte:
celsius_values = [25.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]

plt.plot(days, celsius_values)
plt.show()
```

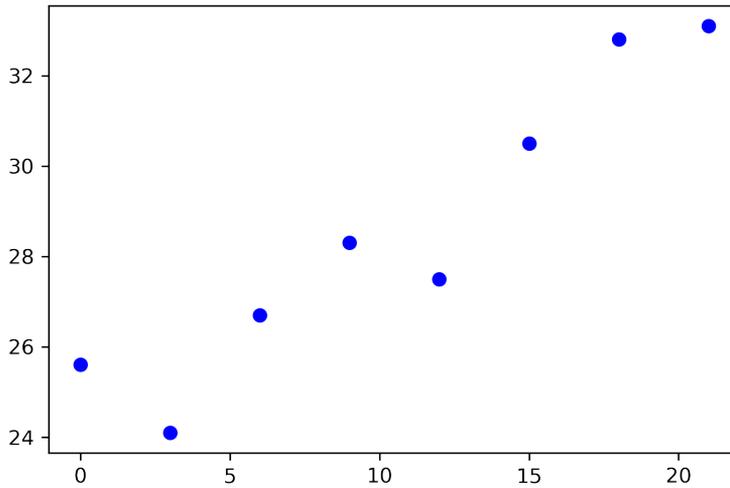
Ausgabe:

```
[0, 3, 6, 9, 12, 15, 18, 21]
```



... und das Ganze wieder mit diskreten Werten:

```
plt.plot(days, celsius_values, 'bo')
plt.show()
```



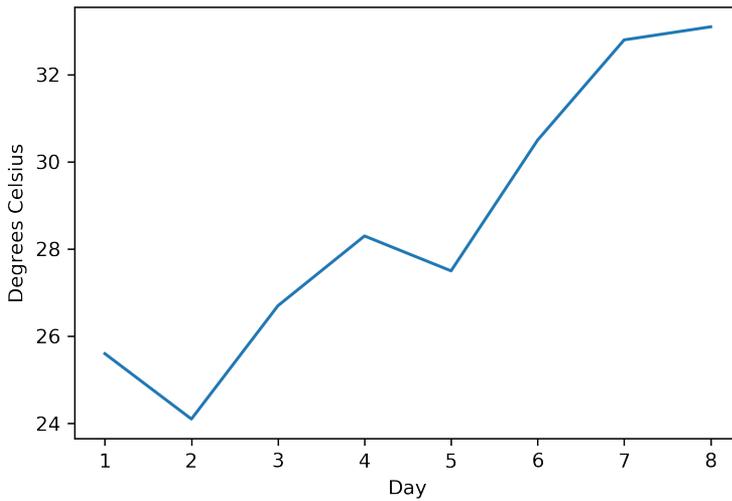
■ 12.3 Bezeichnungen für die Achsen

Wir können das Aussehen unseres Graphen verbessern, indem wir die Achsen mit Bezeichnungen versehen. Dazu benutzen wir die `ylabel`- und `xlabel`-Funktionen von `pyplot`.

```
import matplotlib.pyplot as plt

days = list(range(1,9))
celsius_values = [25.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]

plt.plot(days, celsius_values)
plt.xlabel('Day')
plt.ylabel('Degrees Celsius')
plt.show()
```



Wir können eine beliebige Anzahl von (x, y, fmt)-Gruppen in einer Plot-Funktion spezifizieren. Im folgenden Beispiel benutzen wir zwei verschiedene Listen mit Y-Werten:

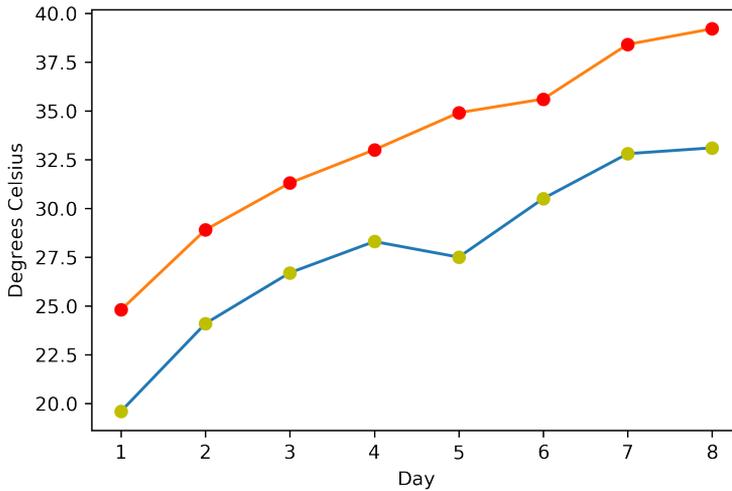
```
import matplotlib.pyplot as plt

days = list(range(1,9))
celsius_min = [19.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]
celsius_max = [24.8, 28.9, 31.3, 33.0, 34.9, 35.6, 38.4, 39.2]

plt.xlabel('Day')
plt.ylabel('Degrees Celsius')

plt.plot(days, celsius_min,
         days, celsius_min, "oy",
         days, celsius_max,
         days, celsius_max, "or")

plt.show()
```



■ 12.4 Abfragen und Ändern des Wertebereichs der Achsen

Mit der Funktion `axis` lässt sich der Wertebereich einer Achse abfragen und ändern. Ruft man `axis` ohne Argumente auf, liefert sie den aktuellen Wertebereich einer Achse zurück:

```
import matplotlib.pyplot as plt

days = list(range(1,9))
celsius_min = [19.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]
celsius_max = [24.8, 28.9, 31.3, 33.0, 34.9, 35.6, 38.4, 39.2]

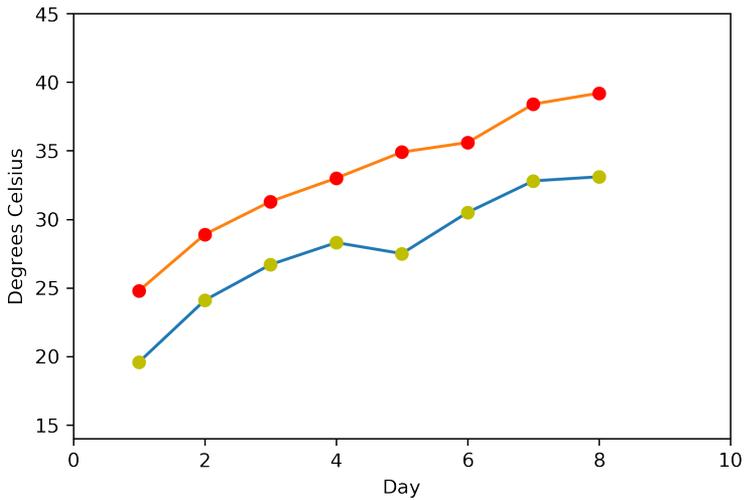
plt.xlabel('Day')
plt.ylabel('Degrees Celsius')

plt.plot(days, celsius_min,
         days, celsius_min, "oy",
         days, celsius_max,
         days, celsius_max, "or")

print("The current limits for the axes are:")
print(plt.axis())
print("We set the axes to the following values:")
xmin, xmax, ymin, ymax = 0, 10, 14, 45
print(xmin, xmax, ymin, ymax)
plt.axis([xmin, xmax, ymin, ymax])
plt.show()
```

Ausgabe:

```
The current limits for the axes are:
(0.6499999999999999, 8.35, 18.62, 40.18)
We set the axes to the following values:
0 10 14 45
```

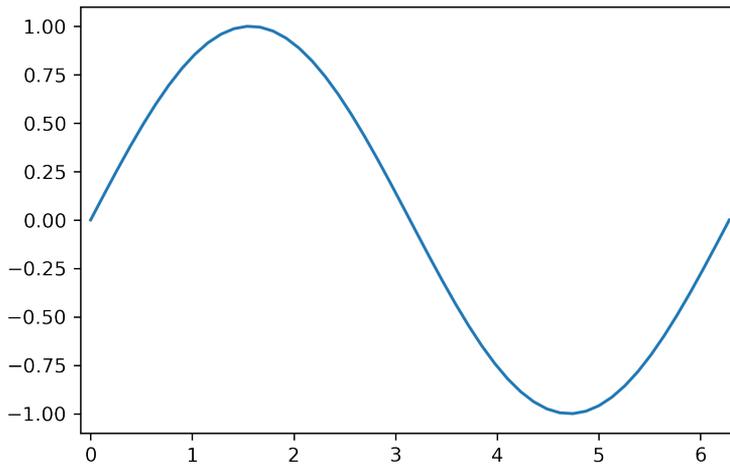


■ 12.5 „linspace“ zur Definition von X-Werten

Im folgenden Beispiel werden wir die NumPy-Funktion `linspace` verwenden. `linspace` wird dazu benutzt, gleichmäßig verteilte Werte innerhalb eines spezifizierten Intervalls zu erzeugen. Wir haben `linspace` ausführlich in unserem NumPy-Kapitel behandelt.

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

X = np.linspace(0, 2 * np.pi, 50, endpoint=True)
F = np.sin(X)
plt.plot(X,F)
startx, endx = -0.1, 2*np.pi + 0.1
starty, endy = -1.1, 1.1
plt.axis([startx, endx, starty, endy])
plt.show()
```

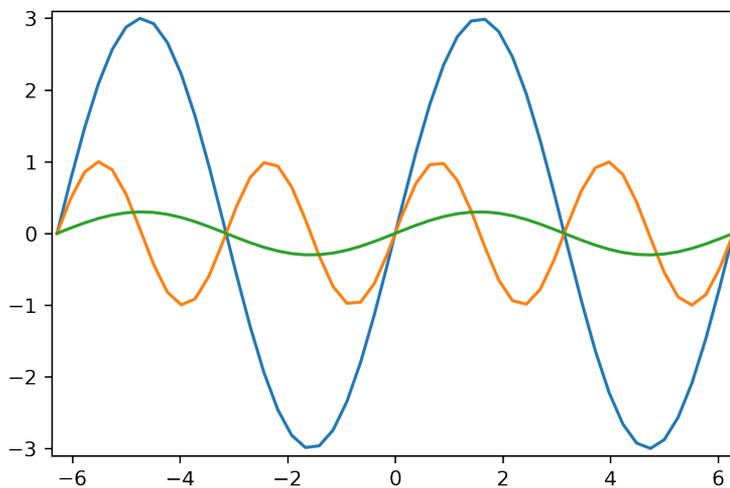


```
import numpy as np
import matplotlib.pyplot as plt

X = np.linspace(-2 * np.pi, 2 * np.pi, 50, endpoint=True)
F1 = 3 * np.sin(X)
F2 = np.sin(2*X)
F3 = 0.3 * np.sin(X)

startx, endx = -2 * np.pi - 0.1, 2*np.pi + 0.1
starty, endy = -3.1, 3.1
plt.axis([startx, endx, starty, endy])

plt.plot(X,F1)
plt.plot(X,F2)
plt.plot(X,F3)
plt.show()
```

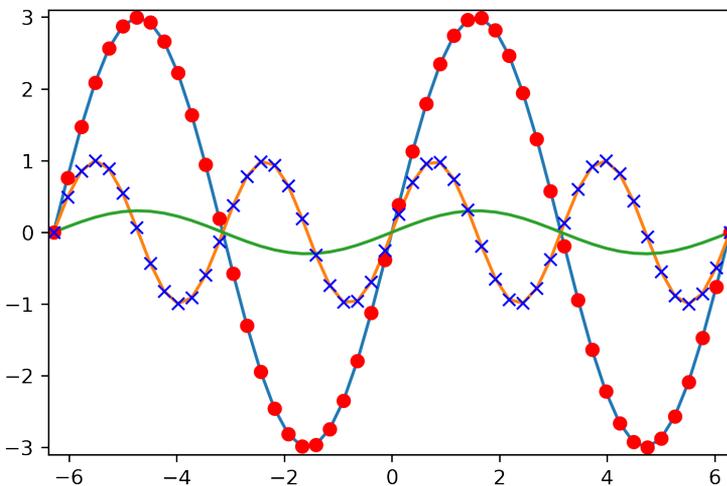


Das nächste Beispiel ist im Prinzip nichts Neues. Wir fügen lediglich zwei weitere Plots mit diskreten Punkten hinzu:

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

X = np.linspace(-2 * np.pi, 2 * np.pi, 50, endpoint=True)
F1 = 3 * np.sin(X)
F2 = np.sin(2*X)
F3 = 0.3 * np.sin(X)
startx, endx = -2 * np.pi - 0.1, 2*np.pi + 0.1
starty, endy = -3.1, 3.1

plt.axis([startx, endx, starty, endy])
plt.plot(X,F1)
plt.plot(X,F2)
plt.plot(X,F3)
plt.plot(X, F1, 'ro')
plt.plot(X, F2, 'bx')
plt.show()
```



■ 12.6 Linienstil ändern

Der Linienstil eines Plots kann durch die Parameter `linestyle` oder `ls` der `plot`-Funktion beeinflusst werden. Sie können auf einen der folgenden Werte gesetzt werden:

'-', '--', '-.', ':', 'None', ''

Wir können mit `linewidth`, wie der Name impliziert, die Linienstärke oder Liniendicke setzen:

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

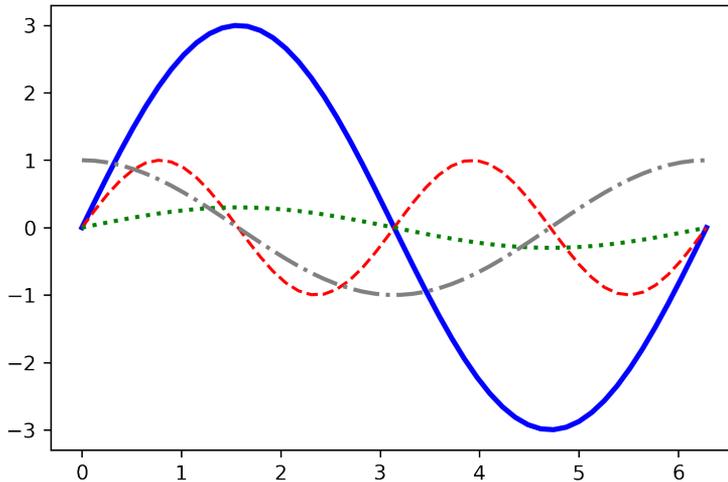
X = np.linspace(0, 2 * np.pi, 50, endpoint=True)
```

```

F1 = 3 * np.sin(X)
F2 = np.sin(2*X)
F3 = 0.3 * np.sin(X)
F4 = np.cos(X)

plt.plot(X, F1, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, F2, color="red", linewidth=1.5, linestyle="--")
plt.plot(X, F3, color="green", linewidth=2, linestyle=":")
plt.plot(X, F4, color="grey", linewidth=2, linestyle="-.")
plt.show()

```



■ 12.7 Flächen einfärben

Mit der pyplot-Funktion `fill_between` ist es möglich, Flächen zwischen Kurven oder Achsen zu schraffieren oder einzufärben. Im folgenden Beispiel füllen wir die Flächen zwischen der X-Achse und dem Graph der Funktion $\sin(2 \cdot X)$:

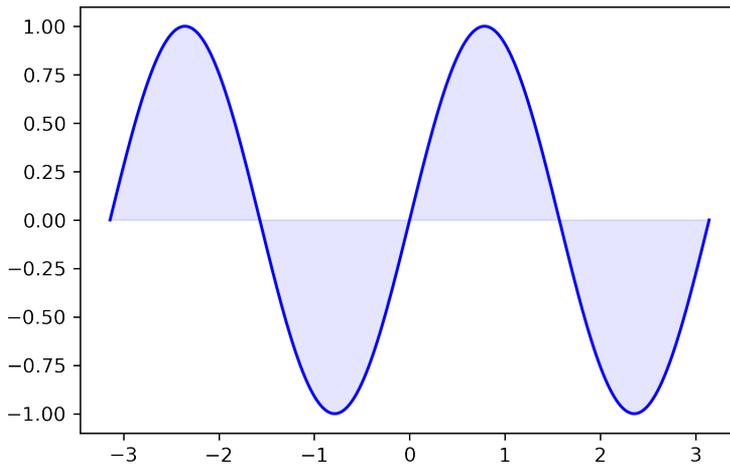
```

import numpy as np
import matplotlib.pyplot as plt

n = 256
X = np.linspace(-np.pi, np.pi, n, endpoint=True)
Y = np.sin(2*X)

plt.plot(X, Y, color='blue', alpha=1.00)
plt.fill_between(X, 0, Y, color='blue', alpha=.1)
plt.show()

```



Die allgemeine Syntax von `fill_between`:

```
fill_between(x, y1, y2=0, where=None, interpolate=False, **kwargs)
```

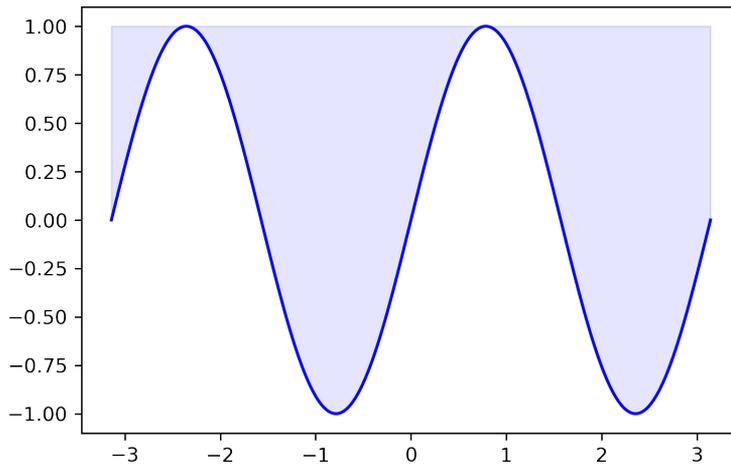
Die Parameter von `fill_between`:

Parameter	Bedeutung
<code>x</code>	Ein Array mit N Elementen mit X-Werten
<code>y1</code>	Ein Array mit N Elementen (oder ein Skalar) von Y-Daten
<code>y2</code>	Ein Array mit N Elementen (oder ein Skalar) von Y-Daten
<code>where</code>	Wenn auf <code>None</code> gesetzt, wird per Default alles gefüllt. Wenn es nicht auf „None“ gesetzt wird, so wird ein numpy boolean-Array erwartet mit N Elementen. Es werden nur dann die Regionen eingefärbt, bei denen <code>where==True</code> ist.
<code>interpolate</code>	Wenn <code>True</code> , so wird zwischen zwei Linien interpoliert, um den genauen Schnittpunkt zu finden. Andernfalls werden die Start- und Endwerte nur als explizite Werte auf der Region erscheinen.
<code>kwargs</code>	Schlüsselwortargumente, die an <code>PolyCollection</code> übergeben werden.

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

n = 256
X = np.linspace(-np.pi,np.pi,n,endpoint=True)
Y = np.sin(2*X)

plt.plot(X, Y, color='blue', alpha=1.00)
plt.fill_between(X, Y, 1, color='blue', alpha=.1)
plt.show()
```

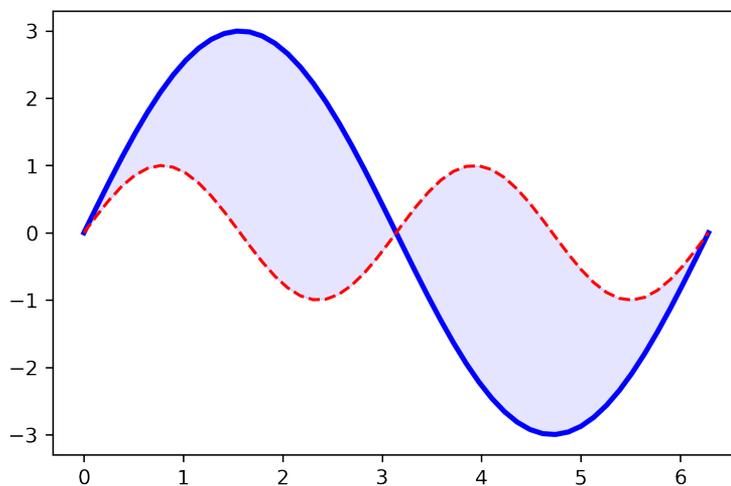


Im nächsten Beispiel füllen wir die Flächen zwischen den Funktionen F1 und F2:

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

```
X = np.linspace(0, 2 * np.pi, 50, endpoint=True)
F1 = 3 * np.sin(X)
F2 = np.sin(2*X)
```

```
plt.plot(X, F1, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, F2, color="red", linewidth=1.5, linestyle="--")
plt.fill_between(X, F1, F2, color='blue', alpha=.1)
plt.show()
```



Stichwortverzeichnis

A

- accumulate 115
- Achsenbeschriftung
 - ändern 186
- Achsenbezeichnung 181
- Achsenbezeichnungen 172
- Achsenteilung 181
- Achsenverschiebung 181
- add_subplot 201
- annotate 194
- arange 51
- Arrays
 - Broadcasting 106
 - concatenate 89
 - Diagonal-Array 68
 - Erzeugen mit Einsen 64
 - Erzeugen mit Nullen 64
 - eye 68
 - flatten 86
 - Identitäts-Array 67
 - identity 67
 - Indizierung 56
 - Konkatenation 89
 - kopieren 65
 - Matrizenmultiplikation 98
 - may_share_memory 60
 - mehrdimensional 43
 - numerische Operationen 95
 - ones 64
 - ones_like 64
 - Operatoren 95
 - ravel 86
 - reshape 88
 - Skalare 95
 - Summenprodukte 104
 - Teilbereichsoperator 57
 - Vergleichsoperatoren 105
 - zeros 64
 - zeros_like 64

- at 118
- atmosphärische Störungen 136
- Attribuierung 35
- Attribute 35
- Ausnahmebehandlung
 - finally 30
- Auswahlen 126

B

- Balkendiagramme 241, 249
- Binarisierung 151
- Binning 299
- bins 300
- Boolesche Indizierung 151
- Boolesche Masken 151
- Boolesche Maskierung 151
- Broadcasting 106

C

- C-zusammenhängend 65
- calendar-Modul 337
- cartesian_choice 132, 133
- C_CONTIGUOUS 65
- choice 126
- coercion 112
- column_stack 91
- concatenate 89
- contour 230
- contourf 230
- count_nonzero 156
- csv-Dateien lesen 157, 283, 284
- csv-Dateien schreiben 157, 285

D

- DataFrame 253, 263
 - Erzeugung aus Series-Objekten 263
 - Index ändern 267
 - reindex 268
 - Spalte in Index wandeln 269

- Spaltennamen 264
 - Spaltenreihenfolge 267
 - Zusammenhang zu Series 263
- DataFrames
- Sortierung 273
 - Spalten einfügen 274
 - Zeilenselektion 270
 - Zugriff auf Spalten 265
 - Zusammenhang mit mehrstufig indizierten Series 310

Datei

- lesen 30
- öffnen 30
- schreiben 30

Datenkonvertierung beim Einlesen 160

datetime 337

datetime-Modul 340

datetime-Objekt in String wandeln 343

datetime.timedelta 342

date.toordinal 338

dateutil 346

dateutil.parser 346

Datum 337

- DIN 5008 337

- Schreibweisen 337

Datum in Landessprache 344

Datumsbereiche erstellen 350

Datumsdifferenzen 342

Diagonal-Array 68

Dimension

- Konkatenation 85

- Reduktion 85

- Summation 85

Dimensionsänderungen 85

DIN 5008 337

Distanzmatrix 111

dot-Produkt 98

dropna 260

dsv-Dateien 283

dsv-Dateien lesen 284

dtype 52, 73

dtypes

- Spaltennamen umbenennen 79

dyadisches Produkt 117

E

echte Zufallszahlen 135

Einfärben von Flächen 178

Eingabeaufforderung

- robust 27

erzwungene Typumwandlung 112

Excel

- openpyxl 287

- xldr 287

Excel-Dateien lesen 287

Excel-Dateien schreiben 287

Excel-Tabelle 254

Exceptions

- eigene generieren 29

eye 68

F

Fancy-Indizierung 153

F_CONTIGUOUS 65

figure 206

- figsize 206

fill_between 178

fillna 261

Filtern fehlender Daten 260

Finalisierungsaktion 29

finally 29, 30

find_interval 127

flatnonzero 156

flatten 86

Fortran-zusammenhängend 65

fully qualified 32

G

Gauss'sche Normalverteilung 137

gca 181

genfromtxt 77, 164

Gestalt eines Arrays 54

getsizeof 46

gezinkter Würfel 128

Gitterlinien 218

gnuplot 167

gridspec 199, 208

H

hist 242

Histogramme 241

I

Identitäts-Array 67

identity 67

import-Anweisung 32

Indizierung

- mehrstufig 307

insert

- DataFrames 274

Instanzzattribut 36

Instanzen 35

Instanziierung 35

IntervallIndex 301

isnull 259

Isolinie 223

K

Kalenderdaten 337

Kartesische Auswahl 132

Kartesische Produkt 132

Klasse 35

– erzeugen 35

Klassenattribut 36

Kommentare 189

Konkatenation

– Pandas 263

Konturen

– gefüllt 228

Konturlinie 223

Konturplot 223, 225

– colors 227

– Farben ändern 227

– individuelle Farben 229

– linestyle 227

– Liniensstil 227

– Schwellen 230

Konvertierung von Daten beim Einlesen 160

Kopieren von Arrays 65

Kuchendiagramme

– Pandas 332

L

Legende

– positionieren 190

Legenden 189

Lesen von Dateien mit NumPy 157

Liniendiagramm

– Pandas 318

Liniendiagramm in Pandas 320

Liniensstil 169

Liniensstil ändern 177

linspace 52

linestyle 177

linewidth 177

loadtxt 78, 159

locale-Modul 344

locale.setlocale 344

logarithmische Darstellung 216

Lotto-Ziehung 131

M

Maschengitter 223

Maskierung 151

math-Modul 32

matplotlib 43, 45, 167

– Beispiel Temperaturwerte 45

– Einfacher Plot 45

Matrizen 43, 73

Matrizenmultiplikation 98

may_share_memory 60

Mehrdimensionale Arrays 54

mehrdimensionale Arrays 43

Mehrstufige Indizes

– Vertauschen 314

mehrstufige Indizierung 307

Mengen

– add 17

Mengenprodukt 132

meshgrid 223, 231

mgrid 231, 235

Module

– eigene Module schreiben 33

– Inhalt 33

– Namensraum 32

– Suchpfad 33

N

Namensraum

– umbenennen 33

NaN 255, 258, 291

– Filtern fehlender Daten 260

– in Dateien 292

– math-Modul 291

– Ursprung 291

– Zusammenhang zu None 260

ndarray.strides 66

newaxis 90

nonzero 154

Norm DIN 5008 337

Normalverteilte Zufallszahlen 137

Normalverteilung 137

notnull 259

NumPy 43

– Akronym 43

– Array-Indizierung 56

– Array-Teilbereichsoperator 57

– Beispiel Dreidimensionales Array 61

– Eindimensionale Arrays 53

– Erzeugen eines einfachen NumPy-Arrays 44

– Erzeugung äquidistanter Intervalle 51

– Erzeugung von Arrays 51

– Gestalt 54

– Matrizen vs. Zweidimensionale Arrays 73

– may_share_memory 60

– Mehrdimensionale Arrays 54

– Nulldimensionale Arrays 53

- numpy.ndarray 53
- Seicherbedarf Arrays 46
- Shape 54
- Zeitvergleich zw. Python-Listen und NumPy-Arrays 49
- Zweidimensionale Arrays 54

O

- ogrid 231, 235
- Olsen-Zeitzone 341
- open() 30
- openpyxl 287
- outer 117

P

- Pandas 43, 253
 - Balkendiagramme 329
 - Dateien einlesen 283
 - Dateien schreiben 283
 - Konkatenation 263
 - Kuchendiagramme 332
 - Liniendiagramm 318, 320
 - Time-Series 347
- Panel data 253
- Parse von Datums- und Zeitstrings 345
- Plot
 - Achsenbezeichnungen 172
 - annotate 194
 - Kommentare 189, 194
 - Legende 189
 - Legende positionieren 190
 - logarithmische Darstellung 216
 - mehrere in einem Diagramm 199
 - Plotbereiche setzen 215
 - speichern 218
- plot 169
- plot-Funktion 168
- plt
 - Alias für matplotlib.pyplot 168
 - xlabel 172
 - ylabel 172
- Population 131
- proleptischer Gregorianischer Kalender 338
- proleptisches Gregorianisches Ordinal 338
- Pseudozufallszahlen 122, 136
- pyplot 168
- pyplot.plot 169

R

- randint 124
- random-Modul 122
- random_sample 130

- random.SystemRandom 122
- ravel 86
- read_csv 284
- record arrays 73
- reduce 116
- reshape 88
- row_stack 91

S

- Säulendiagramm 247
- Säulendiagramme 241
- savefig 218
- savetxt 77, 157
- Schleifen 20
 - for 20
 - Schleifenkörper 20
 - while 20
- Schreiben von Dateien mit NumPy 157
- SciPy 43
- secrets-Modul 122
- secrets.SystemRandom 122
- Seed 136
- Seed-Key 136
- Sekundäre Y-Achse 216
- Series 253, 254
 - apply 257
 - dropna 260
 - Erstellung aus Dictionary 258
 - fillna 261
 - Filterung mit Booleschem Array 256
 - Indizierung 256
 - isnull 259
 - Konkatenation 263
 - notnull 259
 - Vergleich mit NumPy 254
 - Zusammenhang mit Dictionaries 258
- set 17
- setlocale 344
- sets
 - add 17
 - Operationen auf sets 17
- set_xscale 216
- set_yscale 216
- Shape eines Arrays 54
- Skalenteilung 181
- Spaltennamen
 - DataFrame 264
- Spaltenreihenfolge 267
- Spaltenzugriff
 - DataFrames 265
- spines 181
- squeeze 102

stack 312
Startwert einer Zufallsfolge 136
Statistik 121
Stichproben 126, 130
strftime 343
Strichproben 131
strides 66
strptime 345
strukturierte Arrays 73, 75
– Ein- und Ausgabe 77
Subplot
– add_subplot 201
subplot 199
Summenprodukte 104
Synthetische Verkaufszahlen 141

T

Teilbereichsoperator 13, 57
– Listen 13
– NumPy-Arrays 57
tensorielles Produkt 117
Tesseract 85
tile 92
time-Klasse des datetime-Moduls 339
time-Modul 337
Time-Series 347
timedelta 342
timeit 49
Timer 49
to_csv 285
tofile 161
toordinal 338
transpose 155
type coercion 112

U

Ufuncs 112
– accumulate 115
– at 118
– outer 117
– reduce 116
– Rückgabewert 114
Uhrzeiten 337
universelle Funktionen 112
unstack 311
Unterdiagramm 212
urandom 122

V

Variable
– freie 25
– globale 25
– lokale 25
Vertauschen mehrstufiger Indizes 314

W

Wahrscheinlichkeit 121
weighted_choice 127, 128
where 154
Würfel, gezinkt 128
Würfelsimulation 126

X

xlabel 172
xldr 287
xticks 182
– Beschriftung ändern 184
– get_xticklabels 187
– Schriftgröße verändern 187

Y

Y-Achse
– sekundäre 216
ylabel 172
yticks 182

Z

Zeilenselektion
– DataFrame 270
Zeitdifferenzen 342
Zeitmessung 49
Zeitreihen 347
– arithmetische Operationen 348
Zeitreihen 347
Zufallsintervalle 127
Zufallsstichproben 130
Zufallswert 136
Zufallszahlen 122, 136
– echte 135
– kryptografisch stark 122
– normalverteilt 137
Zufallszahlengenerator 136
zusammenhängend 65
Zweidimensionale Arrays 54