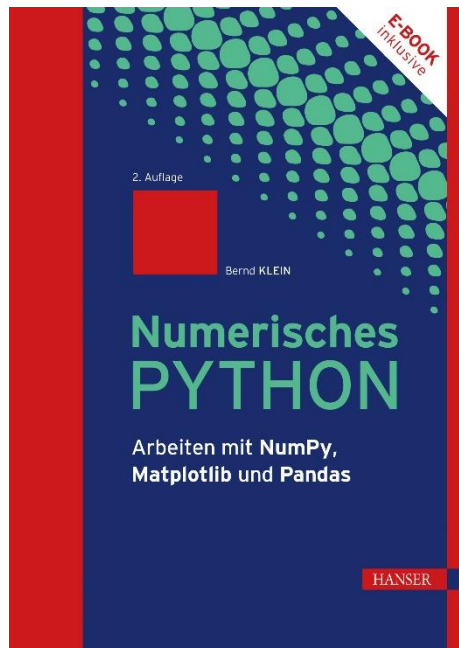


# HANSER



## Leseprobe

zu

## Numerisches Python

von Bernd Klein

Print-ISBN: 978-3-446-47170-2

E-Book-ISBN: 978-3-446-47366-9

E-Pub-ISBN: 978-3-446-47957-9

Weitere Informationen und Bestellungen unter

<https://www.hanser-kundencenter.de/fachbuch/artikel/9783446471702>

sowie im Buchhandel

© Carl Hanser Verlag, München

# Inhalt

<b>Vorwort</b> .....	<b>XV</b>
<b>Danksagung</b> .....	<b>XVI</b>
<b>1 Einleitung</b> .....	<b>1</b>
1.1 Die richtige Wahl .....	1
1.2 Aufbau des Buches .....	2
1.3 Python-Installation .....	2
1.4 Download der Beispiele .....	3
1.5 Anregungen und Kritik .....	3
<b>2 Numerisches Programmieren mit Python</b> .....	<b>4</b>
2.1 Begriffsbestimmung .....	4
2.2 Zusammenhang zwischen Python, NumPy, Matplotlib, SciPy und Pandas .....	5
2.3 Python, eine Alternative zu MATLAB .....	6
<b>Teil I NumPy</b> .....	<b>7</b>
<b>3 NumPy – Einführung</b> .....	<b>9</b>
3.1 Überblick .....	9
3.2 Vergleich NumPy-Datenstrukturen und Python .....	10
3.3 Ein einfaches Beispiel .....	10
3.4 Grafische Darstellung der Werte .....	11
3.5 Speicherbedarf .....	12
3.6 Zeitvergleich zwischen Python-Listen und NumPy-Arrays .....	15
<b>4 Arrays in NumPy erzeugen</b> .....	<b>17</b>
4.1 Erzeugung äquidistanter Intervalle .....	17
4.1.1 arange .....	17
4.1.2 linspace .....	19

4.1.3	Nulldimensionale Arrays in NumPy .....	20
4.1.4	Eindimensionales Array .....	21
4.1.5	Zwei- und mehrdimensionale Arrays .....	21
4.2	Gestalt eines Arrays .....	22
4.3	Indizierung und Teilbereichsoperator .....	23
4.4	Dreidimensionale Arrays .....	28
4.5	Arrays mit Nullen und Einsen .....	31
4.6	Arrays kopieren .....	32
4.6.1	numpy.copy(A) und A.copy() .....	32
4.6.2	Zusammenhängend gespeicherte Arrays .....	32
4.7	Einheitsmatrix .....	34
4.7.1	Die identity-Funktion .....	34
4.7.2	Die eye-Funktion .....	35
4.8	Datentypen .....	36
4.9	Aufgaben .....	38
<b>5</b>	<b>Datentyp-Objekt: dtype .....</b>	<b>40</b>
5.1	dtype .....	40
5.2	Strukturierte Arrays .....	42
5.3	Ein- und Ausgabe von strukturierten Arrays .....	45
5.4	Unicode-Strings in Arrays .....	46
5.5	Umbenennen von Spaltennamen .....	47
5.6	Spaltenwerte austauschen .....	47
5.7	Komplexeres Beispiel .....	48
5.8	Aufgaben .....	49
<b>6</b>	<b>Dimensionsänderungen, Konkatenationen, Stapeln .....</b>	<b>51</b>
6.1	Reduktion und Reshape von Arrays .....	51
6.1.1	flatten .....	51
6.1.2	ravel .....	52
6.1.3	Unterschiede zwischen ravel und flatten .....	53
6.1.4	reshape .....	54
6.2	Weitere Dimensionen hinzufügen .....	55
6.3	Konkatenation von Arrays .....	56
6.4	Vektoren stapeln .....	58
6.4.1	stack-Funktion .....	58
6.4.2	dstack-Funktion .....	60
6.4.3	vstack .....	63
6.4.4	hstack .....	64
6.5	„Fliesen“ mit tile .....	66

<b>7</b>	<b>Numerische Operationen auf NumPy-Arrays</b>	<b>68</b>
7.1	Operatoren und Skalare	68
7.2	Arithmetische Operationen auf zwei Arrays	70
7.3	Matrizenmultiplikation und Skalarprodukt	71
7.3.1	Definition der dot-Funktion	71
7.3.2	Beispiele zur dot-Funktion	71
7.3.3	Das dot-Produkt im dreidimensionalen Fall	72
7.4	Vergleichsoperatoren	78
7.5	Logische Operatoren	79
7.6	Broadcasting	79
7.6.1	Zeilenweises Broadcasting	80
7.6.2	Spaltenweises Broadcasting	83
7.6.3	Broadcasting von zwei eindimensionalen Arrays	85
7.7	Distanzmatrix	86
7.8	ufuncs	87
7.8.1	Anwendung von ufuncs	88
7.8.2	Parameter für Rückgabewerte bei ufuncs	89
7.8.3	accumulate	91
7.8.4	reduce	93
7.8.5	outer	94
7.8.6	at	94
7.9	Aufgaben	95
<b>8</b>	<b>Statistik und Wahrscheinlichkeiten</b>	<b>96</b>
8.1	Einführung	96
8.2	Auf dem random-Modul aufbauende Funktionen	97
8.2.1	Echte Zufallszahlen	97
8.2.2	Erzeugen einer Liste von Zufallszahlen	98
8.2.3	Zufällige Integer-Zahlen	99
8.2.4	Stichproben oder Auswahlen	100
8.2.5	Zufallsintervalle	100
8.2.6	Seed oder Startwert	101
8.2.7	Gewichtete Zufallsauswahl	102
8.2.8	Stichproben mit Python	105
8.2.9	Kartesische Auswahl	106
8.2.10	Kartesisches Produkt	107
8.2.11	Kartesische Auswahl: cartesian_choice	107

8.2.12	Gauss'sche Normalverteilung.....	110
8.2.13	Übung mit Binärsender .....	112
8.3	Das random-Untermodule von NumPy .....	115
8.3.1	Integers und Floats zufällig erzeugen .....	115
8.3.2	<code>numpy.random.choice</code> .....	117
8.3.3	<code>numpy.random.random_sample</code> .....	118
8.4	Synthetische Verkaufszahlen .....	119
8.5	Aufgaben .....	121
<b>9</b>	<b>Boolesche Maskierung und Indizierung .....</b>	<b>123</b>
9.1	Fancy-Indizierung .....	125
9.2	Indizierung mit einem Integer-Array.....	125
9.3	<code>nonzero</code> und <code>where</code> .....	126
9.4	<code>flatnonzero</code> und <code>count_nonzero</code> .....	127
9.5	Aufgaben .....	127
<b>10</b>	<b>Lesen und Schreiben von Daten-Dateien .....</b>	<b>128</b>
10.1	Text-Dateien speichern mit <code>saveetxt</code> .....	128
10.2	Text-Dateien laden mit <code>loadtxt</code> .....	130
10.2.1	<code>loadtxt</code> ohne Parameter.....	130
10.2.2	Spezielle Trenner .....	130
10.2.3	Selektives Einlesen von Spalten .....	131
10.2.4	Datenkonvertierung beim Einlesen .....	131
10.3	<code>tofile</code> .....	133
10.4	<code>fromfile</code> .....	134
10.5	Best Practice, um Daten zu laden und zu speichern .....	135
10.6	Und noch ein anderer Weg: <code>genfromtxt</code> .....	136
10.7	<code>recfromcsv</code> .....	136
<b>Teil II</b>	<b>Matplotlib.....</b>	<b>137</b>
<b>11</b>	<b>Einführung .....</b>	<b>139</b>
11.1	Ein erstes Beispiel.....	140
11.2	Der Formatparameter von <code>pyplot.plot</code> .....	141
11.3	Bezeichnungen für die Achsen.....	143

<b>12</b>	<b>Objekt-Hierarchie</b>	<b>145</b>
12.1	Erzeugung einer Figure und Axes	147
12.2	Achsenbeschriftungen und Titel	148
12.3	Die Plot-Methode	149
12.4	Wertebereiche der Achsen	150
12.5	Plotten mehrerer Funktionen	152
12.6	Streudiagramme	153
12.7	Flächen einfärben	155
<b>13</b>	<b>Mehrfache Plots und Doppelachsen</b>	<b>158</b>
13.1	Mehrere Abbildungen und Achsen	158
<b>14</b>	<b>Gridspec in Matplotlib</b>	<b>166</b>
<b>15</b>	<b>Achsen- und Skalenteilung</b>	<b>174</b>
15.1	Achsenverschiebungen und Achsenbezeichnungen	174
15.2	Achsenbeschriftungen ändern	178
15.3	Justierung der Tick-Beschriftungen	179
<b>16</b>	<b>Legenden und Annotationen</b>	<b>180</b>
16.1	Legende hinzufügen	180
16.2	Annotations/Anmerkungen	183
16.3	Aufgaben	190
<b>17</b>	<b>Konturplots</b>	<b>191</b>
17.1	Erstellen eines Maschengitters	191
17.2	Funktionen auf Meshgrids	193
17.3	contour ohne meshgrid	196
17.4	Linienstil und Farben anpassen	196
17.5	Gefüllte Konturen	198
17.6	Individuelle Farben	199
17.7	Schwellen	200
17.8	Andere Grids	201
17.8.1	Meshgrid genauer	201
17.8.2	mgrid	203
17.8.3	ogrid	203
17.9	imshow	205
17.10	Aufgaben	206

<b>18</b>	<b>Histogramme und Diagramme .....</b>	<b>208</b>
18.1	Histogramme .....	209
18.2	Säulendiagramm .....	213
18.3	Balkendiagramme .....	214
18.4	Gruppierte Balkendiagramme .....	215
18.5	xkcd-Modus .....	219
18.6	Tortendiagramme .....	221
18.7	Stapeldiagramme .....	222
18.8	Aufgaben .....	223
<b>Teil III</b>	<b>Pandas .....</b>	<b>225</b>
<b>19</b>	<b>Pandas: Einführung .....</b>	<b>227</b>
19.1	Datenstrukturen .....	228
19.2	Series .....	228
19.2.1	Indizierung .....	231
19.2.2	pandas.Series.apply .....	232
19.2.3	Zusammenhang zu Dictionaries .....	232
19.3	NaN – Fehlende Daten .....	233
19.3.1	Die Methoden <code>isnull()</code> und <code>notnull()</code> .....	234
19.3.2	Zusammenhang zwischen NaN und None .....	235
19.3.3	Fehlende Daten filtern .....	236
19.3.4	Fehlende Daten auffüllen .....	236
19.4	Aufgaben .....	237
<b>20</b>	<b>DataFrame .....</b>	<b>238</b>
20.1	Zusammenhang zu Series .....	238
20.2	Manipulation der Spaltennamen .....	239
20.3	DataFrames aus Dictionaries .....	240
20.4	Zugriff auf Spalten .....	241
20.5	Selektion von Zeilen .....	242
20.5.1	<code>loc</code> .....	242
20.5.2	<code>query</code> .....	243
20.6	Modifikation von DataFrames .....	245
20.6.1	Spalten einfügen .....	245
20.6.2	Spalten austauschen .....	249
20.6.3	Zeilen austauschen .....	250
20.6.4	Einzelne Werte mittels <code>at</code> und <code>iat</code> ändern .....	250

20.7	Index ändern .....	251
20.7.1	Umsortierung der Spalten und des Index .....	252
20.7.2	Spalten umbenennen .....	253
20.7.3	Spalte in Index umfunktionieren .....	253
20.8	Summen und kumulative Summen .....	254
20.9	Sortierung .....	257
20.10	DataFrame und verschachtelte Dictionaries .....	258
20.11	Aufgaben .....	259
<b>21</b>	<b>Dateiverarbeitung .....</b>	<b>261</b>
21.1	DSV-/CSV-Dateien .....	261
21.1.1	CSV- und DSV-Dateien lesen .....	262
21.1.2	Schreiben von CSV-Dateien .....	264
21.2	Lesen und Schreiben von JSON-Dateien .....	269
21.3	Lesen und Schreiben von Excel-Dateien .....	269
21.4	Aufgaben .....	270
<b>22</b>	<b>Pandas: groupby .....</b>	<b>272</b>
22.1	groupby mit Series .....	272
22.2	Arbeitsweise von groupby .....	274
22.3	groupby mit DataFrames .....	276
22.3.1	groupby mit Funktionen .....	277
22.3.2	Weitere Anwendungen zu groupby .....	279
22.4	Aufgaben .....	283
<b>23</b>	<b>Pivot-Tabellen .....</b>	<b>285</b>
23.1	Pivot-Funktion in Pandas .....	285
23.2	Pivot-Aufruf ohne Werte für values .....	288
23.3	Pivoting auf den Titanic-Daten .....	289
23.4	Aufgaben .....	291
<b>24</b>	<b>Umgang mit NaN .....</b>	<b>292</b>
24.1	nan in Python .....	292
24.2	NaN in Pandas .....	293
24.2.1	Beispiel mit NaNs .....	295
24.3	dropna() verwenden .....	297
24.4	Aufgaben .....	298



<b>25</b>	<b>Binning</b>	<b>299</b>
25.1	Einführung	299
25.2	Binning mit Pandas	302
25.2.1	Binning mit cut	302
25.2.2	Erzeugen eines IntervalIndex-Objektes	303
25.2.3	Mehr zu pd.cut	304
25.2.4	Categorical	305
25.2.5	Binnings mit Labels	305
<b>26</b>	<b>Mehrstufige Indizierung</b>	<b>306</b>
26.1	Einführung	306
26.2	Mehrstufig indizierte Series-Objekte	307
26.3	Alternative Möglichkeiten	307
26.4	Zugriffsmöglichkeiten	308
26.5	Dreistufige Indizes	311
26.6	Zusammenhang zu DataFrames	312
26.6.1	Der harte direkte Weg	312
26.6.2	unstack und stack	313
26.7	Vertauschen mehrstufiger Indizes	316
26.8	Aufgaben	317
<b>27</b>	<b>Datenvisualisierung mit Pandas</b>	<b>319</b>
27.1	Einführung	319
27.2	Liniendiagramm in Pandas	320
27.2.1	Series	320
27.2.2	DataFrames	322
27.2.3	Sekundärachsen (Twin Axes)	325
27.2.4	Mehrere Y-Achsen	326
27.2.5	Spalten mit Zeichenketten (Strings) in Floats wandeln	327
27.3	Balkendiagramme in Pandas	329
27.3.1	Ein einfaches Beispiel	329
27.3.2	Balkengrafik für die Programmiersprachennutzung	329
27.3.3	Farbgebung einer Balkengrafik	331
27.4	Kuchendiagramme in Pandas	331
27.4.1	Ein einfaches Beispiel	331
27.5	Flächenplot mit area	333
27.6	Aufgaben	334

<b>28</b>	<b>Zeit und Datum</b>	<b>335</b>
28.1	Einführung	335
28.2	Python-Standardmodule für Zeit-Daten	336
28.2.1	Die date-Klasse	336
28.2.2	Die time-Klasse	337
28.3	Die datetime-Klasse	338
28.4	Unterschied zwischen Zeiten	340
28.4.1	Wandlung von datetime-Objekten in Strings	341
28.4.2	Wandlung mit strftime	341
28.5	Ausgabe in Landessprache	342
28.6	datetime-Objekte aus Strings erstellen	343
<b>29</b>	<b>Zeitreihen</b>	<b>345</b>
29.1	Einführung	345
29.2	Zeitreihen und Python	345
29.3	Datumsbereiche erstellen	348
29.4	Datumsbereiche mit Uhrzeiten	350
29.5	Aufgaben	351
<b>Teil IV</b>	<b>Anwendungen</b>	<b>353</b>
<b>30</b>	<b>Techniken der Bildverarbeitung</b>	<b>355</b>
30.1	Einführung	355
30.2	Bilder im misc-Paket	356
30.3	Eigene Bilder	358
30.4	Histogramme der Farbwerte	359
30.5	Bilderausschnitte	360
30.6	Geometrische Transformationen	360
30.7	Filtern	362
30.8	Bilder aufhellen und abtönen	366
30.9	Kachelung	374
30.10	Wasserzeichen	376
30.11	Aufgaben	378
<b>31</b>	<b>Finanzverwaltung mit Pandas</b>	<b>379</b>
31.1	Haushaltsbuch	379
31.1.1	Haushaltsbuch mit CSV-Datei	380
31.1.2	Erzeugen eines Excel-Haushaltsbuches	382
31.1.3	Auswertung des Excel-Haushaltsbuches	384

31.2	Einnahmenüberschussrechnung .....	385
31.2.1	Journaldatei .....	386
31.2.2	Analyse und Visualisierung der Daten .....	387
31.2.3	Steuersummen .....	393

## **Teil V   Lösungen zu den Aufgaben ..... 397**

### **32   Lösungen zu den Aufgaben ..... 399**

32.1	Lösungen zu <a href="#">Kapitel 4 (Arrays in NumPy erzeugen)</a> .....	399
32.2	Lösungen zu <a href="#">Kapitel 5 (Datentyp-Objekt: dtype)</a> .....	400
32.3	Lösungen zu <a href="#">Kapitel 7 (Numerische Operationen auf NumPy-Arrays)</a> .....	403
32.4	Lösungen zu <a href="#">Kapitel 8 (Statistik und Wahrscheinlichkeiten)</a> .....	404
32.5	Lösungen zu <a href="#">Kapitel 9 (Boolesche Maskierung und Indizierung)</a> .....	409
32.6	Lösungen zu <a href="#">Kapitel 13 (Mehrfache Plots und Doppelachsen)</a> .....	410
32.7	Lösungen zu <a href="#">Kapitel 16 (Legenden und Annotationen)</a> .....	412
32.8	Lösungen zu <a href="#">Kapitel 17 (Konturplots)</a> .....	414
32.9	Lösungen zu <a href="#">Kapitel 18 (Histogramme und Diagramme)</a> .....	418
32.10	Lösungen zu <a href="#">Kapitel 19 (Pandas: Einführung)</a> .....	421
32.11	Lösungen zu <a href="#">Kapitel 20 (DataFrame)</a> .....	422
32.12	Lösungen zu <a href="#">Kapitel 21 (Dateiverarbeitung)</a> .....	426
32.13	Lösungen zu <a href="#">Kapitel 23 (Pivot-Tabellen)</a> .....	429
32.14	Lösungen zu <a href="#">Kapitel 22 (Pandas: groupby)</a> .....	431
32.15	Lösungen zu <a href="#">Kapitel 24 (Umgang mit NaN)</a> .....	434
32.16	Lösungen zu <a href="#">Kapitel 26 (Mehrstufige Indizierung)</a> .....	435
32.17	Lösungen zu <a href="#">Kapitel 27 (Datenvisualisierung mit Pandas)</a> .....	439
32.18	Lösungen zu <a href="#">Kapitel 29 (Zeitreihen)</a> .....	441
32.19	Lösungen zu <a href="#">Kapitel 30 (Techniken der Bildverarbeitung)</a> .....	441

### **Stichwortverzeichnis ..... 443**

# 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*

# Danksagung

Zum Schreiben eines Buches benötigt es neben der nötigen Erfahrung und Kompetenz im Fachgebiet vor allem viel Zeit. Zeit außerhalb des üblichen Rahmens. Zeit, die vor allem die Familie mitzutragen hat. Deshalb gilt mein besonderer Dank meiner Frau Karola, die mich während dieser Zeit tatkräftig unterstützt hat.

Außerdem danke ich den zahlreichen Teilnehmerinnen und Teilnehmern an meinen Python-Kursen, die mir geholfen haben, meine didaktischen und fachlichen Kenntnisse kontinuierlich zu verbessern. Ebenso möchte ich den Besucherinnen und Besuchern meiner Online-Tutorials unter [www.python-kurs.eu](http://www.python-kurs.eu) und [www.python-course.eu](http://www.python-course.eu) danken, vor allem jenen, die sich mit konstruktiven Anmerkungen bei mir gemeldet haben. Mein besonderer Dank für die zweite Auflage gilt Herrn Tobias Habermann, der dafür gesorgt hat, dass alle Python-Beispiele dieser Auflage automatisch mittels „Pythontex“ getestet und ausgeführt werden, und der auch ansonsten bei der Fehlerkorrektur sehr hilfreich war.

Zuletzt danke ich auch ganz herzlich dem Hanser Verlag, der dieses Buch nun auch in der zweiten, deutlich erweiterten und farbigen Auflage ermöglicht hat. Vor allem danke ich Frau Brigitte Bauer-Schiewek, Programmplanung Computerbuch, und Frau Kristin Rothe, Lektoratsassistentin Computerbuch, für die kontinuierliche ausgezeichnete Unterstützung. LaTeX ist ein fantastisches System, um Bücher zu schreiben, aber ohne die technische Unterstützung von Herrn Stephan Korell und Frau Irene Weilhart bei besonderen LaTeX-Problemen wäre ich manchmal vielleicht verzweifelt. Herrn Jürgen Dubau danke ich fürs Rektorat.

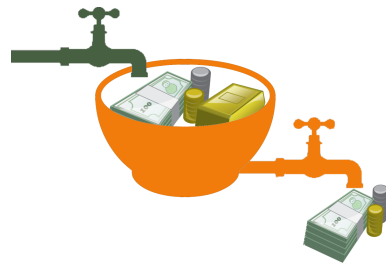
*Bernd Klein, Singen*

In diesem Kapitel führen wir nichts Neues ein, sondern wollen das Erlernte an zwei interessanten Fallbeispielen einüben. Wir haben das Kapitel etwas hochtrabend „Finanzverwaltung“ genannt. In unseren Beispielen geht es jedoch nur um die Implementierung zweier einfacher, aber dennoch überaus wichtiger Konzepte:

- Haushaltsbuch
- Einnahmetüberschussrechnung

In jedem dieser Beispiele geht es um die Verwaltung von Geldern: im ersten Fall für den privaten Bereich

und im zweiten für den kommerziellen Bereich, also mit Mehrwertsteuer. Gemeinsam ist beiden Beispielen auch, dass wir die Daten zur Verarbeitung aus einer Datei beziehen, die wir mit Pandas einlesen.



**Bild 31.1** Geldfluss

## ■ 31.1 Haushaltsbuch

Zuerst geht es nun um einfache Ausgaben- und Einkommenstabellen für den privaten Gebrauch. Einige sagen, wenn man Geld erfolgreich verwalten möchte, muss man die Einnahmen und Ausgaben genau verfolgen. Die Überwachung des Geldflusses ist wichtig, um die eigene finanzielle Situation besser zu verstehen. Man muss genau wissen, wie viel ein- und ausgeht. Dazu bedient man sich üblicherweise eines Haushaltsbuches, in dem alle Ein- und Ausgaben protokolliert sind. Dieses Kapitel will niemanden von der Notwendigkeit überzeugen, dies zu tun. Das Hauptaugenmerk liegt vielmehr auf den Möglichkeiten, die Python und Pandas bieten, um die erforderlichen Tools zu programmieren. Wir zeigen Verfahren, um ein Haushaltsbuch auszuwerten.

### 31.1.1 Haushaltsbuch mit CSV-Datei

Wir beginnen mit einem kleinen Beispiel, das für private Zwecke geeignet ist. Im folgenden Kapitel unseres Pandas-Tutorials wird ein ausführlicheres Beispiel behandelt, welches für kleine Unternehmen geeignet ist.

```
import pandas as pd
ein_aus_df = pd.read_csv("ein_und_ausgaben.csv", sep=";")
print(ein_aus_df[:12]) # Ausgabe der ersten zwölf Zeilen
```

Ausgabe:

	Datum	Beschreibung	Kategorie	Ausgaben
	↪ Einnahmen			
0	2022-07-01	Gehalt Frank	Einkommen	0.00
	↪ 4896.44			
1	2022-07-02	Numerisches Python	Kultur, Bildung	29.00
	↪ 0.00			
2	2022-07-04	Supermarkt	Essen und Getränke	132.40
	↪ 0.00			
3	2022-07-04	Miete	Miete	1267.00
	↪ 0.00			
4	2022-07-04	Gehalt Laura	Einkommen	0.00
	↪ 4910.14			
5	2022-07-04	Strom	Betriebskosten	87.34
	↪ 0.00			
6	2022-07-08	Wasser und Abwasser	Betriebskosten	60.56
	↪ 0.00			
7	2022-07-10	Fitnessstudio, Sarah	Gesundheit und Sport	19.00
	↪ 0.00			
8	2022-07-11	Bank	Kredittilgung	1287.43
	↪ 0.00			
9	2022-07-12	LeGourmet Restaurant	Restaurants und Hotels	145.00
	↪ 0.00			
10	2022-07-13	Supermarkt	Essen und Getränke	197.42
	↪ 0.00			
11	2022-07-13	Pizzeria Pulcinella	Restaurants und Hotels	60.00
	↪ 0.00			

Durch Lesen der CSV-Datei haben wir ein DataFrame-Objekt erstellt. Was können wir damit machen oder mit anderen Worten: Welche Informationen interessieren Frank und Sarah? Natürlich interessieren sie sich für den Kontostand. Sie wollen wissen, wie hoch das Gesamteinkommen war, und sie wollen die Summe aller Ausgaben sehen. Die Salden ihrer Ausgaben und Einnahmen können leicht berechnet werden, indem die Funktion `sum` auf das DataFrame-Objekt `ein_aus_df[['Ausgaben', 'Einnahmen']]` angewendet wird:

```
print(ein_aus_df[['Ausgaben', 'Einnahmen']].sum())
```

Ausgabe:

```
Ausgaben      7660.44
Einnahmen     19613.16
dtype: float64
```

Welche weiteren Informationen möchten Sie aus den Daten gewinnen? Sie könnten daran interessiert sein, die Ausgaben nach den verschiedenen Kategorien zusammengefasst zu sehen. Dies lässt sich mittels `groupby` und `sum` bewerkstelligen:

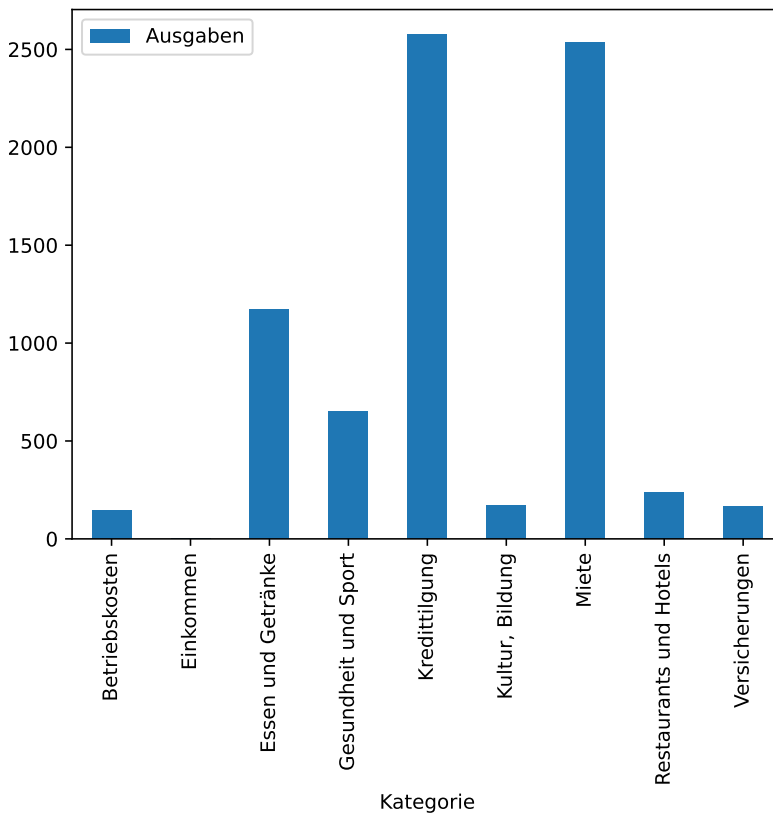
```
category_sums = ein_aus_df.groupby("Kategorie").sum()
print(category_sums)
```

*Ausgabe:*

	Ausgaben	Einnahmen
Kategorie		
Betriebskosten	147.90	0.00
Einkommen	0.00	19613.16
Essen und Getränke	1174.61	0.00
Gesundheit und Sport	650.18	0.00
Kredittilgung	2574.86	0.00
Kultur, Bildung	173.00	0.00
Miete	2534.00	0.00
Restaurants und Hotels	238.00	0.00
Versicherungen	167.89	0.00

Man kann sich die Ein- und Ausgaben auch als Balkendiagramme ausgeben lassen:

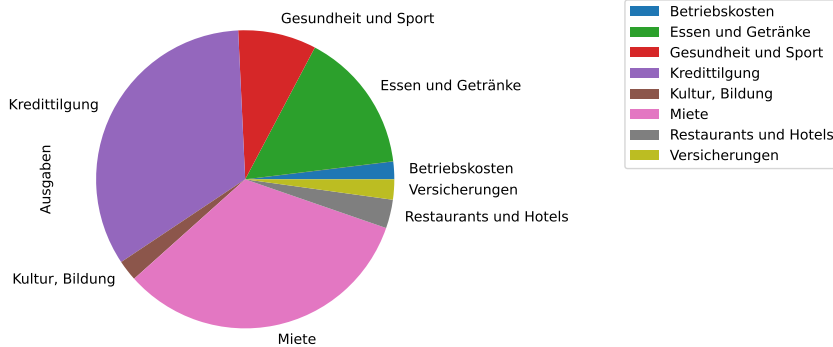
```
ax = category_sums.plot.bar(y="Ausgaben", rot=90)
```





Am besten eignen sich in diesem Falle Tortendiagramme:

```
ax = category_sums.plot.pie(y="Ausgaben")
ax.legend(loc="upper left", bbox_to_anchor=(1.5, 1))
```



### 31.1.2 Erzeugen eines Excel-Haushaltsbuches

Wenn man sich vorstellt, dass man die ganze Zeit Kategorienamen wie „Essen und Getränke“ oder „Restaurants und Hotels“ eingeben muss, kann man sich leicht vorstellen, dass dies zum einen sehr umständlich und zum anderen sehr fehleranfällig ist, da sich leicht Tippfehler im Ausgaben- und Einnahmenjournal einschleichen können.

Es ist daher eine gute Idee, Nummern (Kontonummern) für die Kategorien zu verwenden. Gleichzeitig möchte man aber auch die Kategorien in textueller Form erhalten. Wir wollen nun aus den vorigen Daten eine Excel-Datei erzeugen, die zwei Tabellen (engl. sheets) besitzt. Die erste Tabelle wird die Zuordnung der Kategorien und der Kontonummern enthalten. Die folgenden Kategorien mit Kontonummern sind in unserem Beispiel verfügbar:

Kategorie	Kontonummer
Kredittilgung	200
Versicherungen	201
Essen und Getränke	202
Kultur, Bildung	203
Transport	204
Gesundheit und Sport	205
Haushalt und Dienstleistungen	206
Kleidung	207
Kommunikationskosten	208
Restaurants und Hotels	209
Betriebskosten	210
andere Ausgaben	211
Einkommen	400

Wir können dies als Python-Dictionary implementieren, das Kategorien in Kontonummern abbildet:

```
category2account = {"Kredittilgung": "200",
                    "Versicherungen": "201",
                    "Essen und Getränke": "202",
                    "Kultur, Bildung": "203",
                    "Transport": "204",
                    "Gesundheit und Sport": "205",
                    "Haushalt und Dienstleistungen": "206",
                    "Kleidung": "207",
                    "Kommunikationskosten": "208",
                    "Restaurants und Hotels": "209",
                    "Betriebskosten": "210",
                    "Miete": "211",
                    "andere Ausgaben": "220",
                    "Einkommen": "400"}
```

Im nächsten Schritt werden wir eine Spalte mit den Kontonummern in unser Data-Frame-Objekt `ein_aus_df` einfügen. Dazu formulieren wir zuerst eine Funktion `category2account_func`, die Kategorien in Kontonummern wandelt. Diese Funktion benutzen wir dann in der Methode `apply`, angewendet auf die Spalte `Kategorie`:

```
def category2account_func(category):
    if category in category2account:
        return category2account[category]
    else:
        # doesn't exist
        return '000'

acc = ein_aus_df['Kategorie'].apply(category2account_func)
# Anhängen der 'Konto'-Spalte ans Ende:
ein_aus_df['Konto'] = acc

# Einfügen als Spalte mit Index 2:
ein_aus_df.insert(loc=2,
                  column='Konto',
                  value=acc)
# Löschen der 'Kategorie'-Spalte:
ein_aus_df.drop('Kategorie', axis=1, inplace=True)
print(ein_aus_df[:7])
```

*Ausgabe:*

	Datum	Beschreibung Konto	Ausgaben	Einnahmen
0	2022-07-01	Gehalt Frank 400	0.00	4896.44
1	2022-07-02	Numerisches Python 203	29.00	0.00
2	2022-07-04	Supermarkt 202	132.40	0.00
3	2022-07-04	Miete 211	1267.00	0.00
4	2022-07-04	Gehalt Laura 400	0.00	4910.14
5	2022-07-04	Strom 210	87.34	0.00
6	2022-07-08	Wasser und Abwasser 210	60.56	0.00

Wir werden dieses DataFrame-Objekt ebenfalls in unserer Excel-Datei als eine separate Tabelle abspeichern. Diese Excel-Datei wird somit zwei Tabellen enthalten: eine mit dem Namen „Journal“ und die andere mit der Zuordnung von Konto zu Kategoriennamen unter dem Namen „Kontonummern“.

```
account_numbers = pd.Series(category2account)
account_numbers.name = "Kontonummern"

with pd.ExcelWriter('ein_und_ausgaben.xlsx') as writer:
    account_numbers.to_excel(writer, "Kontonummern")
    ein_aus_df.to_excel(writer, "Journal")
```

### 31.1.3 Auswertung des Excel-Haushaltsbuches

Jetzt kann man diese Excel-Datei als Grundlage für das Haushaltsbuch nehmen. Wir müssen unser obiges Programm nun entsprechend anpassen. Zunächst lesen wir die Excel-Datei ein:

```
import pandas as pd

with pd.ExcelFile('ein_und_ausgaben.xlsx') as xl:
    print(xl.sheet_names)
    accounts = xl.parse('Kontonummern', index_col=1)
    journal = xl.parse('Journal', index_col=0)

kat = journal.Konto.replace(accounts.index,
                             accounts.values)
journal['Kategorie'] = kat
print(accounts[:5])
print(journal[:5])
```

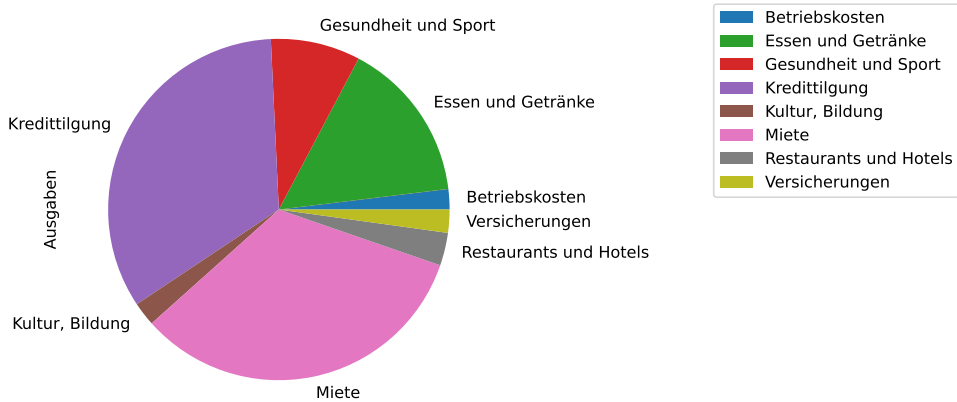
*Ausgabe:*

```

                                Unnamed: 0
Kontonummern
200                                Kredittilgung
201                                Versicherungen
202                Essen und Getränke
203                Kultur, Bildung
204                Transport
      Datum      Beschreibung  Konto  Ausgaben  Einnahmen
↳ Kategorie
0  2022-07-01      Gehalt Frank    400         0.0    4896.44
↳ Einkommen
1  2022-07-02  Numerisches Python    203         29.0         0.00    Kultur,
↳ Bildung
2  2022-07-04          Supermarkt    202        132.4         0.00    Essen und
↳ Getränke
3  2022-07-04             Miete    211       1267.0         0.00
↳ Miete
4  2022-07-04      Gehalt Laura    400         0.0    4910.14
↳ Einkommen
```

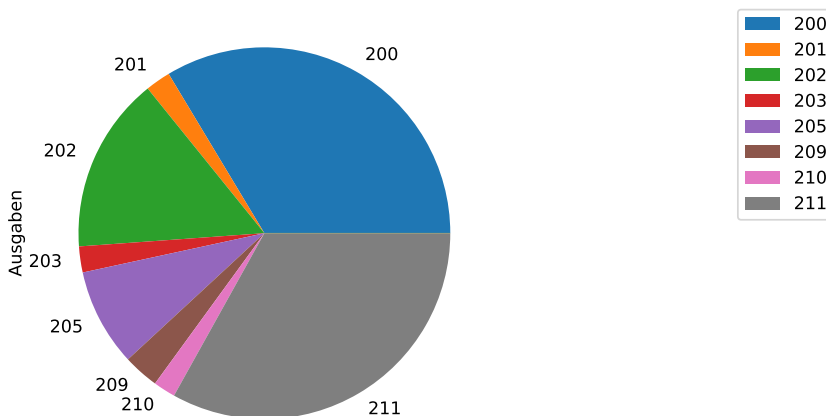
Wir erzeugen nun ein Tortendiagramm aus den Daten nach den Kategorien gruppiert:

```
category_sums = journal.groupby('Kategorie').sum()
ax = category_sums.plot.pie(y="Ausgaben")
ax.legend(loc="upper left", bbox_to_anchor=(1.5, 1))
```



Nach Kontonummern können wir ähnlich leicht gruppieren:

```
konten_summen = journal.groupby('Konto').sum()
ax = konten_summen.plot.pie(y="Ausgaben")
ax.legend(loc="upper left", bbox_to_anchor=(1.5, 1))
```



## ■ 31.2 Einnahmenüberschussrechnung

Bei der in Deutschland benutzten Einnahmenüberschussrechnung (EÜR), in Österreich als Einnahmen-Ausgaben-Rechnung (E/A-Rechnung) bekannt, handelt es sich um eine vereinfachte Gewinnermittlungsmethode, die so vom Gesetz vorgegeben und für bestimmte

Berufsgruppen anerkannt ist. Die Einnahmenüberschussrechnung ist sowohl in Deutschland als auch in Österreich im § 4 Abs. 3 des jeweiligen Einkommensteuergesetzes (EStG) geregelt. Alle Steuerpflichtigen, die nicht zur doppelten Buchführung verpflichtet sind, dürfen dieses vereinfachte Verfahren zur Gewinnermittlung verwenden.

Seit Anfang 2013 wurde sie auch in der Schweiz mit dem neuen Rechnungslegungsrecht eingeführt. Seitdem sind kleine Unternehmen auch in der Schweiz nicht mehr dazu verpflichtet, ihre Bücher doppelt zu führen. Stattdessen reicht nun eine einfache Buchhaltung, auch Milchbüchli-Rechnung genannt, was der deutschen EÜR entspricht.

Bei der EÜR gilt das Zufluss- und Abflussprinzip, was bedeutet, dass lediglich die Einnahmen bzw. Ausgaben zu berücksichtigen sind, die in dem entsprechenden Wirtschaftsjahr vereinnahmt bzw. gezahlt wurden. Bestandsveränderungen bleiben unberücksichtigt. Auch wenn hier und an anderen Stellen immer von „vereinfachter“ oder von „einfacher“ Gewinnermittlungsmethode die Rede ist, so darf das nicht darüber hinwegtäuschen, dass auch bei dieser Methode viele gesetzlichen Regeln zu beachten sind. Im Rahmen dieses Buches kann natürlich nicht auf die komplexe Materie eingegangen werden. Die hier vorgestellten Verfahren können für die eigene EÜR verwendet werden, aber es kann keine Garantie auf die Richtigkeit gegeben werden.

Wir sind hauptsächlich daran interessiert, die verschiedenen Möglichkeiten vorzustellen, wie Pandas und Python zur Einnahmeüberschussrechnung benutzt werden können. Wir zeigen, wie es mit Python möglich ist, den Geldfluss zu überwachen und zu visualisieren. Auf diese Weise kann man sich einen besseren Überblick über die finanzielle Situation des Betriebes oder der selbständigen Tätigkeit verschaffen. Die hier vorgestellten Algorithmen können auch steuerlich eingesetzt werden. Aber seien Sie gewarnt: Dies ist eine sehr allgemeine Behandlung der Angelegenheit und muss an die tatsächliche Steuersituation angepasst werden, d. h. es kann keine Gewähr für die steuerliche Richtigkeit übernommen werden.

### 31.2.1 Journaldatei

Als Grundlage für unsere Einnahmeüberschussrechnung dient ein Excel-Dokument mit zwei Tabellen: eine mit den laufenden Posten, also den Einnahmen und Ausgaben in chronologischer Reihenfolge. Diese Tabelle bezeichnen wir als Journal. Die zweite Tabelle enthält die Zuordnungen von Kontonummern zu den Kontobezeichnungen. Im Folgenden lesen wir diese Excel-Datei in zwei DataFrame-Objekte ein:

```
import pandas as pd

with pd.ExcelFile("einnahme_ueber_2022.xlsx") as xl:
    konto_beschreibung = xl.parse("kontenplan",
                                   index_col=0)
    journal = xl.parse("journal",
                       index_col=0)

print(f"Die ersten 15 Zeilen von journal:\n{journal[:12]}")
print(f"Kontobeschreibung:\n{konto_beschreibung}")
```

*Ausgabe:*

Die ersten 15 Zeilen von journal:

Datum	Kto	DokuNummer	Beschreibung	StSatz	Brutto
2022-04-02	4402	8983233038	Zurkan, Köln	19	4105.98
2022-04-02	2010	57550799	Bengelmann, Souvenirs	19	-1890.00
2022-04-02	2200	14989004	Gehälter	0	-17478.23
2022-04-02	2500	12766279	Tankstelle, Benzin	19	-89.40
2022-04-02	4400	3733462359	EnergyCom, Hamburg	19	4663.54
2022-04-02	4402	7526058231	Enoigo, Strasbourg	19	2412.82
2022-04-05	4402	1157284466	Qbooks, Frankfurt	7	2631.42
2022-04-05	4402	7009463592	Qbooks, Köln	7	3628.45
2022-04-05	2020	68433353	Jamdon, Kleider	19	-1900.00
2022-04-05	2010	53353169	Outleg, Souvenirs	19	-2200.00
2022-04-09	4402	4775929332	Drupa AG, Freiburg	19	4751.66
2022-04-09	2100	10759896	Gebäudeversicherung	0	-467.00

Kontobeschreibung:

Kto	Beschreibung
4400	Standort München
4401	Standort Frankfurt
4402	Standort Berlin
2010	Souvenirs
2020	Kleider
2030	Andere Artikel
2050	Bücher
2100	Versicherungen
2200	Gehälter
2300	Kredite
2400	Hotels
2500	Benzin
2600	Telekommunikation
2610	internet

### 31.2.2 Analyse und Visualisierung der Daten

Es gibt viele Möglichkeiten, diese Daten zu analysieren. Wir können zum Beispiel alle Konten zusammenfassen:

```
konten_summen = journal[["Kto", "Brutto"]].groupby("Kto").sum()
print(konten_summen)
```

*Ausgabe:*

Kto	Brutto
2010	-4090.00
2020	-10500.80
2030	-1350.00
2050	-900.00
2100	-612.00
2200	-69912.92
2300	-18791.92

```

2400  -1597.10
2500  -89.40
2600  -492.48
2610  -561.00
4400  37771.84
4401  69610.35
4402  61593.99

```

Nun wollen wir die Daten mittels Kreisdiagrammen, auch Kuchendiagramme genannt, visualisieren. Im Englischen bezeichnet man sie als „pie charts“. Nun haben wir aber ein kleines Problem: Kreisdiagramme dürfen keine negativen Werte enthalten. Dies lässt sich jedoch schnell beheben. Wir können die Konten in Einnahmen- und Ausgabenkonten aufteilen. Das entspricht natürlich auch mehr dem, was wir wirklich sehen wollen.

Wir erstellen zunächst ein DataFrame mit den Einnahmen und eines mit den Ausgaben. Nach der Erzeugung der Ausgabensummen multiplizieren wir das Ergebnis mit -1, um die Werte positiv zu machen:

```

einnahmen = konten_summen[konten_summen["Brutto"] > 0]
ausgaben = konten_summen[konten_summen["Brutto"] < 0] * -1

print(f"---- Einnahmen ----\n{einnahmen}")
print(f"---- Ausgaben ----\n{ausgaben}")

```

*Ausgabe:*

```

---- Einnahmen ----
      Brutto
Kto
4400  37771.84
4401  69610.35
4402  61593.99
---- Ausgaben ----
      Brutto
Kto
2010  4090.00
2020  10500.80
2030  1350.00
2050   900.00
2100   612.00
2200  69912.92
2300  18791.92
2400   1597.10
2500    89.40
2600   492.48
2610   561.00

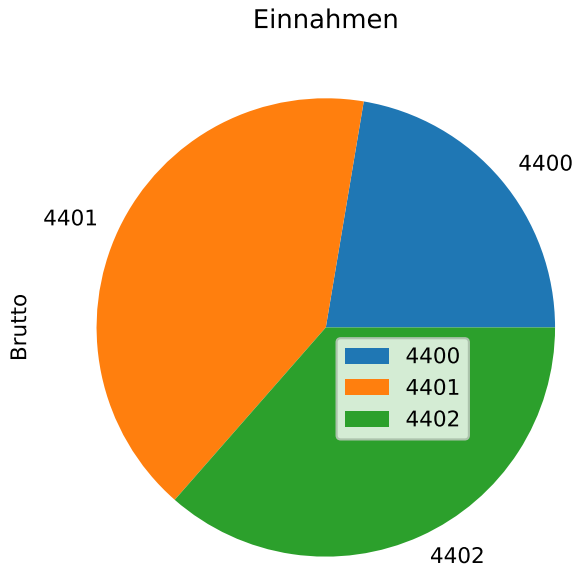
```

Nun erzeugen wir ein Tortendiagramm mit den Einnahmen:

```

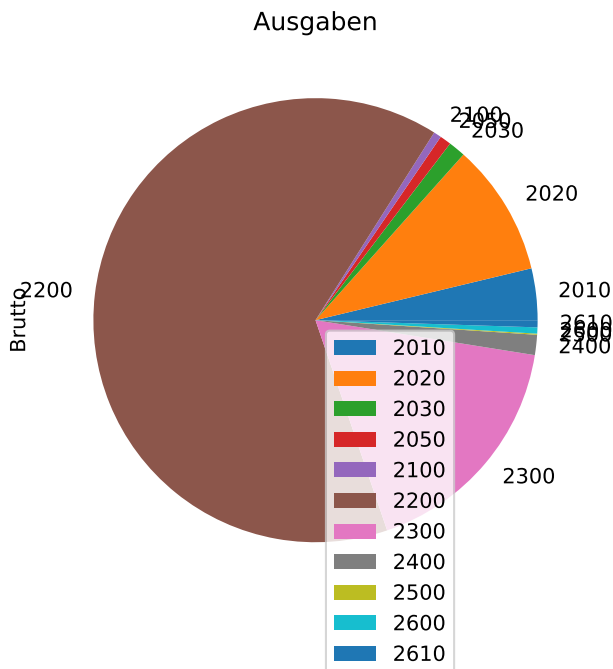
ax = einnahmen.plot(y='Brutto',
                    title='Einnahmen',
                    kind="pie")
ax.legend(bbox_to_anchor=(0.5, 0.5),
          loc="upper left")

```



Nun erzeugen wir analog das entsprechende Tortendiagramm für die Ausgaben.

```
ax = ausgaben.plot(y='Brutto',
                  title='Ausgaben',
                  kind="pie")
ax.legend(bbox_to_anchor=(0.5, 0.5),
          loc="upper left")
```



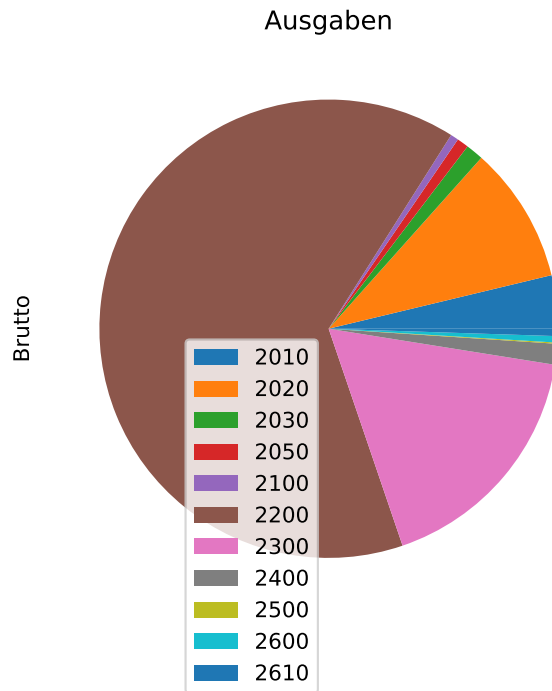


Es gibt zwei Möglichkeiten zu verhindern, dass sich die Labels nicht überlappen. Die erste besteht einfach darin, dass man sie nicht ausgibt. Man hat ja immer noch die nötige Information durch die Farbzuzuordnung in der Legende.

Nun erzeugen wir analog die Labels nur in der Legende:

```
ax = ausgaben.plot(y='Brutto',
                   title='Ausgaben',
                   kind="pie",
                   labels=[''] * len(ausgaben))

ax.legend(bbox_to_anchor=(0.5, 0.5),
          labels=ausgaben.index)
```

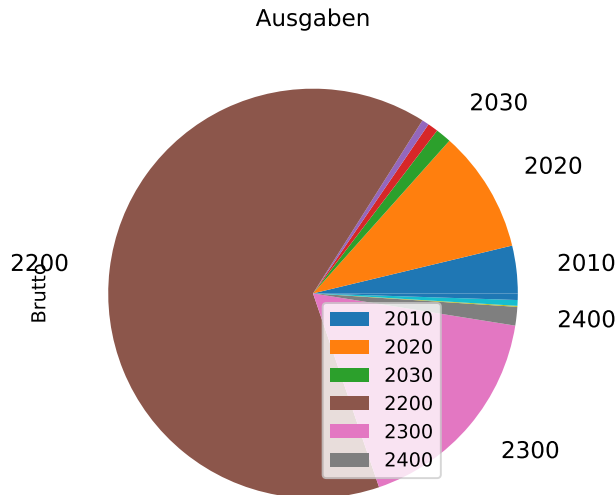


Bei der zweiten Möglichkeit unterdrücken wir die Ausgabe, wenn die Werte unterhalb einer bestimmten Schwelle liegen. Dazu kopieren wir das DataFrame und setzen dort die entsprechenden Labels im Index auf einen leeren String. Zuerst wird der DataFrame `ausgaben` verwendet, um die prozentualen Anteile jeder Kategorie zu berechnen. Die Werte werden dann der Variable `ausg_norm` zugewiesen. Anschließend wird der Index von `ausg_norm` basierend auf den Werten in der Spalte `Brutto` geändert, sodass Kategorien mit einem Anteil von weniger als 1 % nicht im Diagramm dargestellt werden. Die Lambda-Funktion wird verwendet, um den Index von `ausg_norm` zu ändern. Wenn der Bruttoanteil einer Kategorie kleiner als 1 ist, wird der Index-Wert auf einen leeren String (") gesetzt, ansonsten wird der Index-Wert unverändert beibehalten.

Das kreisförmige Diagramm wird dann mit der `plot`-Methode von Pandas erstellt, indem `kind="pie"` angegeben wird. Die Ausgabenkategorien werden als Labels in den Abschnitten des Diagramms angezeigt. Die `labeldistance`-Option legt die Entfernung der Labels vom Zentrum des Diagramms fest, während die `textprops`-Option verwendet wird, um die Schriftgröße der Labels anzupassen. Eine Legende wird mit der `legend`-Methode von Matplotlib erstellt und auf der linken oberen Seite des Diagramms platziert.

```
ausg_norm = ausgaben * 100 / ausgaben.sum()
print(ausg_norm)
ausg_norm.index = ausg_norm.index.map(lambda x: '' \
                                     if ausg_norm.loc[x, 'Brutto'] < 1 else x)

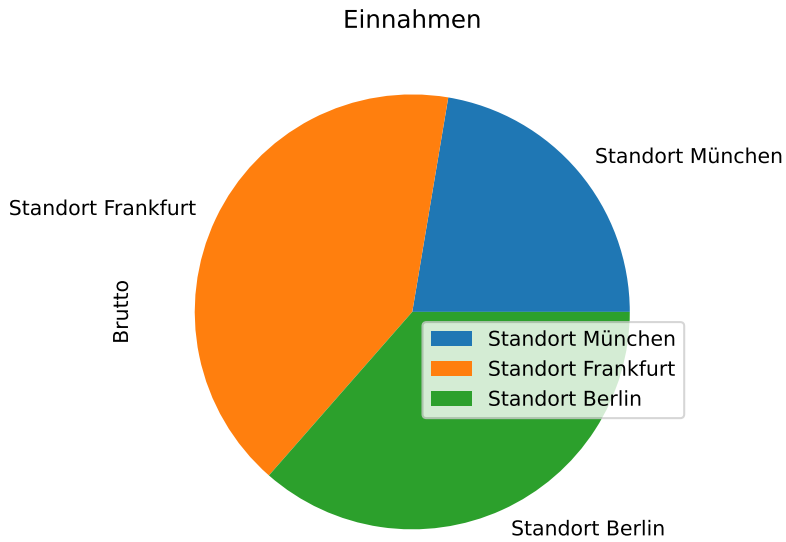
ax = ausg_norm.plot(y='Brutto',
                    title='Ausgaben',
                    labeldistance=1.2,
                    textprops={'fontsize': 12},
                    kind="pie")
ax.legend(bbox_to_anchor=(0.5, 0.5),
          loc="upper left")
```



Nun mit den Kontennamen statt den Kontonummern für die Einnahmen:

```
beschreibungen = konto_beschreibung["Beschreibung"].loc[einnahmen.index]

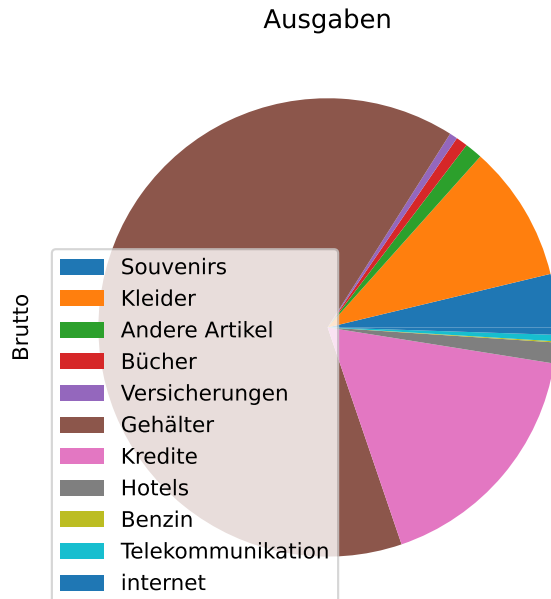
ax = einnahmen.plot(y='Brutto',
                    title='Einnahmen',
                    labels=beschreibungen,
                    kind="pie")
ax.legend(bbox_to_anchor=(0.5, 0.5),
          labels=beschreibungen,
          loc="upper left")
```



```
beschreibungen = konto_beschreibung["Beschreibung"].loc[ausgaben.index]
```

```
ax = ausgaben.plot(y='Brutto',
                  title='Ausgaben',
                  kind="pie",
                  labels=[''] * len(ausgaben))
```

```
ax.legend(loc="lower left",
         labels=beschreibungen)
```

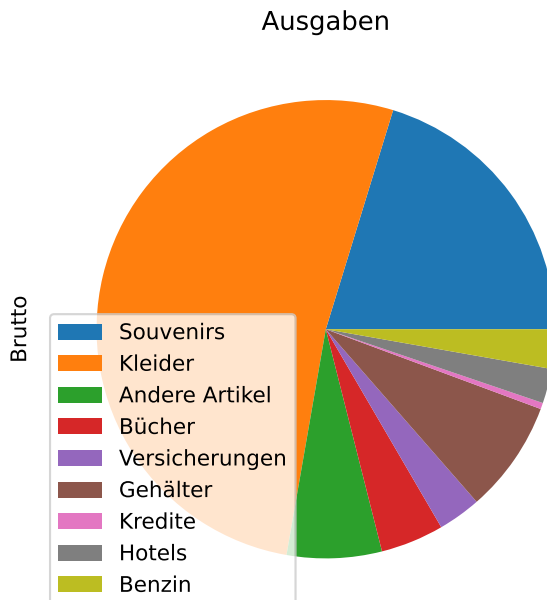


Die Ausgabenkonten mit den niedrigen Salden sind nur sehr schlecht in unseren Plots erkennbar. Eine Möglichkeit, dies zu verbessern, besteht darin, die großen Posten wie Gehälter (2200) und Kredite (2300) in der Ausgabe zu unterdrücken:

```
beschreibungen = konto_beschreibung["Beschreibung"].loc[ausgaben.index]

ax = ausgaben.drop([2200, 2300]).plot(y='Brutto',
                                     title='Ausgaben',
                                     kind="pie",
                                     labels=[''] * len(ausgaben))

ax.legend(loc="lower left",
          labels=beschreibungen)
```



### 31.2.3 Steuersummen

Wir wollen nun die Steuersummen berechnen. Wir benutzen dazu die Spalte StSatz, die den Steuersatz enthält. Im Folgenden definieren wir nun eine Funktion `steuersummen`, die die Mehrwertsteuersummen nach Steuersätzen aus einem Journal-DataFrame berechnet:

```
import pandas as pd

def steuersummen(journal_df, monate=None):
    """ Liefert ein DataFrame mit den Umsätzen und Steuersätzen
    zurück. Wird monate eine Zahl oder Liste übergeben, werden
    nur die Umsätze der entsprechenden Monate berücksichtigt.
    Beispiel: steuersummen(df, monate=[3, 6]) bedeutet nur die
    Monate 3 (März) und 6 (Juni)"""
```

```

if monate:
    if isinstance(monate, int):
        monate_cond = journal_df.index.month == monate
    elif isinstance(monate, (list, tuple)):
        monate_cond = journal_df.index.month.isin(monate)
    positive = journal_df["Brutto"] > 0
    umsatzsteuern = journal_df[positive & monate_cond]
    negative = journal_df["Brutto"] < 0
    vorsteuern = journal_df[negative & monate_cond]
else:
    umsatzsteuern = journal_df[journal_df["Brutto"] > 0]
    vorsteuern = journal_df[journal_df["Brutto"] < 0]

umsatzsteuern = umsatzsteuern[["StSatz", "Brutto"]].groupby("StSatz").sum()
umsatzsteuern.rename(columns={"Brutto": "Umsaetze brutto"},
                     inplace=True)
umsatzsteuern.index.name = 'Steuerrate'

vorsteuern = vorsteuern[["StSatz", "Brutto"]].groupby("StSatz").sum()
vorsteuern.rename(columns={"Brutto": "Ausgaben brutto"},
                  inplace=True)
vorsteuern.index.name = 'Steuerrate'

steuern = pd.concat([vorsteuern, umsatzsteuern], axis=1)
steuern.insert(1,
               column="Vorsteuer",
               value=(steuern["Ausgaben brutto"] * steuern.index /
                      ↪ 100).round(2))
steuern.insert(3,
               column="Umsatzsteuer",
               value=(steuern["Umsaetze brutto"] * steuern.index /
                      ↪ 100).round(2))

return steuern.fillna(0)

```

Wir lesen die Daten wieder aus unserer Excel-Datei ein und berechnen dann die Steuersummen nach Steuersätzen mit der Funktion `steuersummen` und `journal` als Argument:

```

with pd.ExcelFile("einnahme_ueber_2022.xlsx") as xl:
    konto_beschreibung = xl.parse("kontenplan",
                                   index_col=0)
    journal = xl.parse("journal",
                       index_col=0,
                       )
    sts = steuersummen(journal)
    print(sts)

```

*Ausgabe:*

	Ausgaben brutto	Vorsteuer	Umsaetze brutto	Umsatzsteuer
Steuerrate				
0	-90102.20	-0.00	8334.43	0.00
7	-3847.10	-269.30	11240.71	786.85
19	-14948.32	-2840.18	149401.04	28386.20

Die Steuersummen für den Monat Mai berechnet man wie folgt:

```
sts = steuersummen(journal, monate=5)
print(sts)
```

*Ausgabe:*

	Ausgaben brutto	Vorsteuer	Umsaetze brutto	Umsatzsteuer
Steuerrate				
0	-22411.53	-0.00	0.00	0.00
7	-900.00	-63.00	0.00	0.00
19	-145.00	-27.55	31328.98	5952.51

Jetzt berechnen wir die Steuern für März und April:

```
sts = steuersummen(journal, monate=[3, 4])
print(sts)
```

*Ausgabe:*

	Ausgaben brutto	Vorsteuer	Umsaetze brutto	Umsatzsteuer
Steuerrate				
0	-22878.53	-0.00	1854.96	0.00
7	-1239.10	-86.74	6259.87	438.19
19	-8480.20	-1611.24	37061.01	7041.59



# Stichwortverzeichnis

## A

- accumulate 91
- Achsenbeschriftung
  - ändern 178
- Achsenbezeichnung 143, 174
- Achsenteilung 174
- Achsenverschiebung 174
- annotate 186
- Annotationen 180
- arange 17
- Arrays
  - Broadcasting 79
  - concatenate 56
  - Diagonal-Array 35
  - Erzeugen mit Einsen 31
  - Erzeugen mit Nullen 31
  - eye 35
  - flatten 51
  - Identitäts-Array 34
  - identity 34
  - Indizierung 23
  - Konkatenation 56
  - kopieren 32
  - Matrizenmultiplikation 71
  - may\_share\_memory 27
  - mehrdimensional 9
  - numerische Operationen 68
  - ones 31
  - ones\_like 31
  - Operatoren 68
  - ravel 52
  - reshape 54
  - Skalare 68
  - Summenprodukte 77
  - Teilbereichsoperator 24
  - Vergleichsoperatoren 78
  - zeros 31
  - zeros\_like 31

- assign → DataFrame
- at 94, → DataFrame
- atmosphärische Störungen 98
- Auswählen 100
- axes 145

## B

- Balkendiagramme 208, 214
  - gruppiert 215
- Bildverarbeitung 355
  - Bildausschnitte 360
  - Geometrische Transformationen 360
- Binarisierung 123
- Binning 299
- binning
  - IntervallIndex 303
  - pd.cut 302
- bins 300
- Boolesche Indizierung 123
- Boolesche Masken 123
- Boolesche Maskierung 123
- Broadcasting 79
  - meshgrid erzeugen 204

## C

- calendar-Modul 335
- cartesian\_choice 106, 107
- C\_CONTIGUOUS 32
- choice 100
- column\_stack 64, 65
- concatenate 56
- contour 200
- contourf 200
- converters 267, 429
- count 280
- count\_nonzero 127
- csv-Dateien lesen 128, 261, 262
- csv-Dateien schreiben 128, 264



cut 302

C-zusammenhängend 32

## D

DataFrame 228, 238

– assign 248

– at 250

– drop 282

– Einzelne Werte ändern 250

– Erzeugung aus Series-Objekten 238

– iat 250

– Index ändern 251

– Index umsortieren 252

– insert 247

– loc 242, 245

– Modifikation 245

– read\_csv 263

– reindex 252

– round 290

– Series nach 312

– Sortierung 257

– Spalte in Index wandeln 253

– Spalten 245

– Spalten aus Dictionary auswählen 251

– Spalten austauschen 249

– Spalten einfügen 245

– Spalten einfügen mit loc 248

– Spalten umbenennen 253

– Spalten umsortieren 252

– Spaltennamen 239

– Spaltenreihenfolge 252

– stack 313

– Werte ändern 245

– Zeilen austauschen 250

– Zeilen einfügen mit loc 248

– Zeilenselektion 242

– Zugriff auf Spalten 241

– Zugriff auf Zeilen mittels loc 242

– Zusammenhang mit mehrstufig  
indizierten Series 312

– Zusammenhang zu Series 238

Datenkonvertierung beim Einlesen 131

datetime 335

datetime-Modul 338

datetime-Objekt in String wandeln 341

datetime.timedelta 340

date.toordinal 336

dateutil 344

dateutil.parser 344

Datum 335

– DIN 5008 335

– Schreibweisen 335

Datum in Landessprache 342

Datumsbereiche erstellen 348

Datumsdifferenzen 340

Diagonal-Array 35

Dimension

– Konkatenation 51

– Reduktion 51

– Summation 51

Dimensionsänderungen 51

DIN 5008 335

Distanzmatrix 86

dot-Produkt 71

drop 282

dropna 236

dstack 60

dsv-Dateien 261

– lesen 262

dtype 18, 40

– Spaltennamen umbenennen 47

dyadisches Produkt 94

## E

echte Zufallszahlen 97

Einfärben von Flächen 155

Einheitsmatrix 34

Einnahmen-Ausgaben-Rechnung 385

Einnahmeüberschussrechnung 379, 385

EÜR 385

Excel

– openpyxl 269

– xldr 269

Excel-Dateien lesen 269

Excel-Dateien schreiben 269

Excel-Tabelle 228

eye 35

## F

Fancy-Indizierung 125

Farbpalette 165

F\_CONTIGUOUS 32

figure 145

– Breite ändern 180

– Länge ändern 180

– set\_figheight 180

– set\_figwidth 180

fill\_between 155

fillna 236

Filtern fehlender Daten 236

Finanzverwaltung 379

find\_interval 100

flatnonzero 127

flatten 51  
Fortran-zusammenhängend 32

## G

Gauss'sche Normalverteilung 110  
genfromtxt 45, 136  
Gestalt eines Arrays 22  
getsizeof 13  
get\_xticklabels 175  
get\_yticklabels 175  
gezinkter Würfel 103  
gnuplot 139, 145  
gridspec 158  
groupby  
– count 280  
– size 281  
Gruppierte Balkendiagramme 215

## H

Haushaltsbuch 379  
Hierarchischer Index 306  
– dreistufig 311  
hist 209  
Histogramm 208, 209  
Höhenlinie 191  
hstack 64

## I

iat → DataFrame  
Identitäts-Array 34  
identity 34  
iloc 242  
imshow 205, 356  
Index  
– dreistufig 311  
– hierarchisch 306, 307  
– mehrstufig 307  
index\_col 264  
Indizierung  
– mehrstufig 306, 307  
insert → DataFrame  
– DataFrames 245  
IntervalIndex 303  
isnull 234  
Isolinie 191

## J

Journaldatei 386  
JSON-Dateien, Lesen und Schreiben 269

## K

Kalenderdaten 335  
Kartesische Auswahl 106

Kartesisches Produkt 107  
Kommentare 180  
Konkatenation  
– Pandas 238  
Konturen  
– gefüllt 198  
Konturlinie 191  
Konturplot 191, 194  
– colors 196  
– Farben ändern 196  
– individuelle Farben 199  
– linestyles 196  
– Liniensstil 196  
– Schwellen 200  
Konvertierung von Daten beim Einlesen 131  
Kopieren von Arrays 32  
Kuchendiagramm  
– Pandas 331

## L

Legenden 180  
– positionieren 183  
Lesen von Dateien mit NumPy 128  
Liniendiagramm  
– Pandas 320, 322  
Liniensstil 141  
linspace 19, 152  
loadtxt 45, 130  
loc 242, 245, → DataFrame  
locale-Modul 342  
locale.setlocale 342  
Lotto-Ziehung 105

## M

Maschengitter 191  
Maskierung 123  
MATLAB 145  
matplotlib 9, 11, 137, 145  
– Beispiel Temperaturwerte 11  
– Einfacher Plot 11  
Matrizen 9, 40  
Matrizenmultiplikation 71  
may\_share\_memory 27  
mehrdimensionale Arrays 9, 21  
mehrfache Plots 158  
mehrstufige Indizes  
– Vertauschen 316  
mehrstufige Indizierung 306, 307  
Mengenprodukt 107  
Meshgrid  
– erzeugen 204  
meshgrid 191, 201

mgrid 201, 203  
 Milchbüchli-Rechnung 386  
 misc 356  
 MultiIndex 307  
 MultiIndex.from\_product 308  
 MultiIndex.from\_tuples 307

## N

NaN 230, 233, 292  
 – Filtern fehlender Daten 236  
 – in Dateien 293  
 – math-Modul 292  
 – Ursprung 292  
 – Zusammenhang zu None 235  
 ndarray.strides 34  
 newaxis 55  
 nonzero 126  
 Norm DIN 5008 335  
 Normalverteilte Zufallszahlen 110  
 Normalverteilung 110  
 notnull 234  
 np.concatenate 56  
 np.mesgrid 191  
 np.stack 58, 372  
 NumPy 9  
 – Akronym 9  
 – Array-Indizierung 23  
 – Array-Teilbereichsoperator 24  
 – Beispiel dreidimensionales Array 28  
 – Eindimensionale Arrays 21  
 – Erzeugen eines einfachen NumPy-Arrays 11  
 – Erzeugung äquidistanter Intervalle 17  
 – Erzeugung von Arrays 17  
 – Gestalt 22  
 – Matrizen vs. zweidimensionale Arrays 40  
 – may\_share\_memory 27  
 – mehrdimensionale Arrays 21  
 – Nulldimensionale Arrays 20  
 – numpy.ndarray 20  
 – Shape 22  
 – Speicherbedarf Arrays 12  
 – Zeitvergleich zw. Python-Listen und NumPy-Arrays 15  
 – Zweidimensionale Arrays 21

## O

ogrid 201, 203  
 Olsen-Zeitzone 339  
 openpyxl 269  
 outer 94

## P

Pandas 9, 225  
 – Balkendiagramme 329  
 – Dateien einlesen 261  
 – Dateien schreiben 261  
 – Konkatenation 238  
 – Kuchendiagramme 331  
 – Liniendiagramm 320, 322  
 – Time-Series 345  
 Panel data 227  
 Parsen von Datums- und Zeitstrings 343  
 pd.cut 302  
 pd.IntervallIndex 303  
 pie chart 221  
 Pivot-Tabellen 285  
 Plot 141  
 – Achsenbezeichnung 143  
 – Anmerkungen 183  
 – annotate 186  
 – Annotationen 180  
 – Annotations 183  
 – Kommentare 180, 183, 186  
 – Legende 180  
 – Legende positionieren 183  
 – mehrere in einem Diagramm 158  
 Plot-Funktion 140  
 plots  
 – einfärben 164  
 plt  
 – Alias für matplotlib.pyplot 140  
 – xlabel 143  
 – ylabel 143  
 plt.imshow 205  
 plt-Schnittstelle 145  
 PNG-Bilder 357  
 Population 105  
 proleptischer Gregorianischer Kalender 336  
 proleptisches Gregorianisches Ordinal 336  
 Pseudozufallszahlen 97, 101  
 Punktdiagramme 153  
 pyplot 140  
 pyplot.plot 141

## Q

query 243  
 – Leerzeichen in Spaltennamen 245  
 – Variablen 244

## R

randint 99  
 random-Modul 97  
 random\_sample 118

random.SystemRandom 97  
ravel 52  
read\_csv 262, 429  
– converters 267, 429  
record arrays 40  
reduce 93  
reindex → DataFrame  
reshape 54, 56  
round 290  
row\_stack 63, 64

## S

Säulendiagramm 208, 213  
– gestapeltes 222  
savefig 145  
savetxt 45, 128  
Schreiben von Dateien mit NumPy 128  
SciPy 9  
scipy 356  
secrets-Modul 97  
secrets.SystemRandom 97  
Seed 101  
Seed-Key 101  
Series 228  
– apply 232  
– DataFrame nach 312  
– dropna 236  
– Erstellung aus Dictionary 232  
– fillna 236  
– Filterung mit Booleschem Array 231  
– Index 230  
– Indizierung 231  
– isnull 234  
– Konkatenation 238  
– notnull 234  
– swaplevel 316  
– unstack 313  
– Values 230  
– Vergleich mit NumPy 229  
– Zusammenhang mit Dictionaries 232  
setlocale 342  
set\_xticklabels 175  
set\_yticklabels 175  
Shape eines Arrays 22  
sharex 159  
sharey 159  
size 281  
Skalenteilung 174  
Slicing 23, 24  
– Bildausschnitte 360  
Spalten aus Dictionary auswählen →  
DataFrame

Spaltennamen  
– DataFrame 239  
Spaltenreihenfolge → DataFrame  
Spaltenzugriff  
– DataFrames 241  
Spiegelung eines Bildes 442  
spine 174  
squeeze 75  
stack 58, 313, 314, 372  
stack plots 222  
Stapeldiagramm 222  
Startwert einer Zufallsfolge 101  
Statistik 96  
Stichproben 100, 105  
Streudiagramme 153  
strftime 341  
strides 33, 34  
strptime 343  
strukturierte Arrays 40, 42  
– Ein- und Ausgabe 45  
subplot 158  
– cols 159  
– Parameter 159  
– rows 159  
– sharex 159  
– sharey 159  
– title 163  
Subplots 153  
subplots\_adjust 163  
– Parameters 163  
Summenprodukte 77  
suptitle 163  
swaplevel 316  
Synthetische Verkaufszahlen 119

## T

Teilbereichsoperator 23, 24  
– Bildausschnitte 360  
– NumPy-Arrays 24  
tempfile 135  
TemporaryFile 135  
tensorielles Produkt 94  
Tesseract 51  
Tiefenlinie 191  
tile 66  
timedelta 340  
timeit 15  
time-Klasse des datetime-Moduls 337  
time-Modul 335  
Time-Series 345  
Timer 15  
TIOBE 1, 221

Titanic-Daten 289

to\_csv 264

tofile 133

toordinal 336

Tortendiagramm 221

transpose 126

Tranzparenzwert 357

## U

ufunc 87

– accumulate 91

– reduce 93

ufuncs

– at 94

– outer 94

– Rückgabewert 89

Uhrzeiten 335

universelle Funktionen 87

unstack 313

urandom 97

## V

Vertauschen mehrstufiger Indizes 316

vstack 63, 64

## W

Wahrscheinlichkeit 96

weighted\_choice 100, 102

Weltbevölkerung 222

where 126

Würfel, gezinkt 103

Würfelsimulation 116

## X

xkcd-Modus 219

xlabel 143

xldr 269

xticklabels 175

xticks 175

– get\_xticklabels 179

– Schriftgröße verändern 179

## Y

ylabel 143

yticklabels 175

yticks 175

## Z

Zeilenselektion

– DataFrame 242

Zeitdifferenzen 340

Zeitmessung 15

Zeitreihen 345

– arithmetische Operationen 346

Zufallsintervalle 100

Zufallsstichproben 105

Zufallswert 101

Zufallszahlen 97, 101

– echte 97

– kryptografisch starke 97

– normalverteilt 110

Zufallszahlengenerator 101

zusammenhängend 32

Zweidimensionale Arrays 21