

Inhalt

1	Einleitung	29
----------	-------------------	----

2	Die Programmiersprache Python	37
----------	--------------------------------------	----

2.1	Historie, Konzepte, Einsatzgebiete	37
2.1.1	Geschichte und Entstehung	37
2.1.2	Grundlegende Konzepte	38
2.1.3	Einsatzmöglichkeiten und Stärken	39
2.1.4	Einsatzbeispiele	40
2.2	Die Installation von Python	41
2.2.1	Installation von Anaconda unter Windows	41
2.2.2	Installation von Anaconda unter Linux	42
2.2.3	Installation von Anaconda unter macOS	43
2.3	Drittanbietermodule installieren	43
2.4	Die Verwendung von Python	44

TEIL I Einstieg in Python

3	Erste Schritte im interaktiven Modus	47
----------	---------------------------------------------	----

3.1	Ganze Zahlen	47
3.2	Gleitkommazahlen	49
3.3	Zeichenketten	50
3.4	Listen	50
3.5	Dictionaries	51
3.6	Variablen	52
3.6.1	Die besondere Bedeutung des Unterstrichs	53
3.6.2	Bezeichner	53
3.7	Logische Ausdrücke	54

- 3.8 Funktionen und Methoden** 56
 - 3.8.1 Funktionen 56
 - 3.8.2 Methoden 57
- 3.9 Bildschirmausgaben** 58
- 3.10 Module** 59

4 Der Weg zum ersten Programm 61

- 4.1 Tippen, kompilieren, testen** 61
 - 4.1.1 Shebang 63
 - 4.1.2 Interne Abläufe 63
- 4.2 Grundstruktur eines Python-Programms** 65
 - 4.2.1 Umbrechen langer Zeilen 66
 - 4.2.2 Zusammenfügen mehrerer Zeilen 67
- 4.3 Das erste Programm** 68
- 4.4 Kommentare** 70
- 4.5 Der Fehlerfall** 71

5 Kontrollstrukturen 73

- 5.1 Fallunterscheidungen** 73
 - 5.1.1 Die if-Anweisung 73
 - 5.1.2 Bedingte Ausdrücke 76
- 5.2 Schleifen** 78
 - 5.2.1 Die while-Schleife 78
 - 5.2.2 Abbruch einer Schleife 79
 - 5.2.3 Erkennen eines Schleifenabbruchs 79
 - 5.2.4 Abbruch eines Schleifendurchlaufs 81
 - 5.2.5 Die for-Schleife 83
- 5.3 Die pass-Anweisung** 85
- 5.4 Zuweisungsausdrücke** 86
 - 5.4.1 Motivation 87
 - 5.4.2 Das Spiel Zahlenraten mit einem Zuweisungsausdruck 88

6	Dateien	89
6.1	Datenströme	89
6.2	Daten aus einer Datei auslesen	90
6.2.1	Eine Datei öffnen und schließen	91
6.2.2	Die with-Anweisung	91
6.2.3	Den Dateiinhalt auslesen	92
6.3	Daten in eine Datei schreiben	95
6.4	Das Dateiojekt erzeugen	96
6.4.1	Die Built-in Funktion open	96
6.4.2	Attribute und Methoden eines Dateiojekts	98
6.4.3	Die Schreib-/Leseposition verändern	99
7	Das Datenmodell	103
7.1	Die Struktur von Instanzen	105
7.1.1	Datentyp	106
7.1.2	Wert	107
7.1.3	Identität	108
7.2	Referenzen löschen	109
7.3	Mutable vs. immutable Datentypen	111
7.3.1	Mutable Datentypen und Seiteneffekte	112
8	Funktionen, Methoden und Attribute	115
8.1	Parameter von Funktionen und Methoden	115
8.1.1	Positionsbezogene Parameter	116
8.1.2	Schlüsselwortparameter	117
8.1.3	Optionale Parameter	117
8.1.4	Reine Schlüsselwortparameter	118
8.2	Attribute	118

9 Informationsquellen zu Python 121

9.1 Die Built-in Function help 121

9.2 Die Onlinedokumentation 122

9.3 PEPs 122

TEIL II Datentypen

10 Basisdatentypen: eine Übersicht 127

10.1 Das Nichts – NoneType 128

10.2 Operatoren 129

 10.2.1 Bindigkeit 130

 10.2.2 Auswertungsreihenfolge 132

 10.2.3 Verkettung von Vergleichen 132

11 Numerische Datentypen 135

11.1 Arithmetische Operatoren 135

 11.1.1 Erweiterte Zuweisungen 136

11.2 Vergleichende Operatoren 137

11.3 Konvertierung zwischen numerischen Datentypen 138

11.4 Ganzzahlen – int 139

 11.4.1 Zahlensysteme 140

 11.4.2 Bit-Operationen 142

 11.4.3 Die Methode bit_length 146

11.5 Gleitkommazahlen – float 146

 11.5.1 Exponentialschreibweise 147

 11.5.2 Genauigkeit 147

 11.5.3 Unendlich und Not a Number 148

11.6 Boolesche Werte – bool 149

 11.6.1 Logische Operatoren 149

11.6.2	Wahrheitswerte nicht-boolescher Datentypen	152
11.6.3	Auswertung logischer Operatoren	154
11.7	Komplexe Zahlen – complex	155

12 Sequenzielle Datentypen 159

12.1	Der Unterschied zwischen Text und Binärdaten	159
12.2	Operationen auf Instanzen sequenzieller Datentypen	161
12.2.1	Auf Elemente prüfen	162
12.2.2	Verkettung	163
12.2.3	Wiederholung	165
12.2.4	Indizierung	166
12.2.5	Slicing	167
12.2.6	Länge einer Sequenz	170
12.2.7	Das kleinste und das größte Element	171
12.2.8	Ein Element suchen	171
12.2.9	Elemente zählen	172
12.3	Listen – list	173
12.3.1	Verändern eines Wertes innerhalb der Liste – Zuweisung mit []	174
12.3.2	Ersetzen von Teillisten und Einfügen neuer Elemente – Zuweisung mit []	174
12.3.3	Elemente und Teillisten löschen – del zusammen mit []	175
12.3.4	Methoden von list-Instanzen	176
12.3.5	Listen sortieren - s.sort([key, reverse])	178
12.3.6	Seiteneffekte	182
12.3.7	List Comprehensions	184
12.4	Unveränderliche Listen – tuple	187
12.4.1	Packing und Unpacking	188
12.4.2	Immutabel heißt nicht zwingend unveränderlich!	190
12.5	Strings – str, bytes, bytearray	190
12.5.1	Steuerzeichen	193
12.5.2	String-Methoden	195
12.5.3	Formatierung von Strings	206
12.5.4	Zeichensätze und Sonderzeichen	217

13 Zuordnungen und Mengen 227

13.1 Dictionary – dict	227
13.1.1 Erzeugen eines Dictionarys	227
13.1.2 Schlüssel und Werte	229
13.1.3 Iteration	230
13.1.4 Operatoren	231
13.1.5 Methoden	234
13.1.6 Dict Comprehensions	240
13.2 Mengen – set und frozenset	241
13.2.1 Erzeugen eines Sets	241
13.2.2 Iteration	243
13.2.3 Operatoren	243
13.2.4 Methoden	249
13.2.5 Veränderliche Mengen – set	250
13.2.6 Unveränderliche Mengen – frozenset	252

14 Collections 255

14.1 Verkettete Dictionarys	255
14.2 Zählen von Häufigkeiten	256
14.3 Dictionarys mit Standardwerten	259
14.4 Doppelt verkettete Listen	260
14.5 Benannte Tupel	261

15 Datum und Zeit 265

15.1 Elementare Zeitfunktionen – time	265
15.1.1 Der Datentyp struct_time	266
15.1.2 Konstanten	267
15.1.3 Funktionen	268
15.2 Objektorientierte Datumsverwaltung – datetime	273
15.2.1 datetime.date	274
15.2.2 datetime.time	275
15.2.3 datetime.datetime	276

15.2.4	<code>datetime.timedelta</code>	278
15.2.5	Operationen für <code>datetime.datetime</code> und <code>datetime.date</code>	281
15.3	Zeitzone – <code>zoneinfo</code>	283
15.3.1	Die IANA-Zeitzone-Datenbank	283
15.3.2	Zeitangaben in lokalen Zeitzone	284
15.3.3	Rechnen mit Zeitangaben in lokalen Zeitzone	285

16 Enumerationen und Flags 289

16.1	Aufzählungstypen – <code>Enum</code>	289
16.2	Aufzählungstypen für Bitmuster – <code>Flag</code>	291
16.3	Ganzzahlige Aufzählungstypen – <code>IntEnum</code>	292

TEIL III Fortgeschrittene Programmiertechniken

17 Funktionen 297

17.1	Definieren einer Funktion	299
17.2	Rückgabewerte	300
17.3	Funktionsobjekte	302
17.4	Optionale Parameter	303
17.5	Schlüsselwortparameter	304
17.6	Beliebige Anzahl von Parametern	305
17.7	Reine Schlüsselwortparameter	307
17.8	Reine Positionsparameter	308
17.9	Unpacking beim Funktionsaufruf	309
17.10	Seiteneffekte	312
17.11	Namensräume	315
17.11.1	Zugriff auf globale Variablen – <code>global</code>	315
17.11.2	Zugriff auf den globalen Namensraum	316
17.11.3	Lokale Funktionen	317

17.11.4	Zugriff auf übergeordnete Namensräume – nonlocal	318
17.11.5	Ungebundene lokale Variablen – eine Stolperfalle	320
17.12	Anonyme Funktionen	321
17.13	Rekursion	322
17.14	Eingebaute Funktionen	323
17.14.1	abs(x)	327
17.14.2	all(iterable)	327
17.14.3	any(iterable)	327
17.14.4	ascii(object)	328
17.14.5	bin(x)	328
17.14.6	bool([x])	328
17.14.7	bytearray([source, encoding, errors])	329
17.14.8	bytes([source, encoding, errors])	330
17.14.9	chr(i)	330
17.14.10	complex([real, imag])	330
17.14.11	dict([source])	331
17.14.12	divmod(a, b)	332
17.14.13	enumerate(iterable[, start])	332
17.14.14	eval(expression, [globals, locals])	332
17.14.15	exec(object, [globals, locals])	333
17.14.16	filter(function, iterable)	334
17.14.17	float([x])	334
17.14.18	format(value, [format_spec])	334
17.14.19	frozenset([iterable])	335
17.14.20	globals()	335
17.14.21	hash(object)	336
17.14.22	help([object])	336
17.14.23	hex(x)	337
17.14.24	id(object)	337
17.14.25	input([prompt])	337
17.14.26	int([x, base])	338
17.14.27	len(s)	338
17.14.28	list([sequence])	338
17.14.29	locals()	339
17.14.30	map(function, [*iterable])	339
17.14.31	max(iterable, {default, key}), max(arg1, arg2, [*args], {key})	340
17.14.32	min(iterable, {default, key}), min(arg1, arg2, [*args], {key})	341
17.14.33	oct(x)	341
17.14.34	ord(c)	342
17.14.35	pow(x, y, [z])	342

17.14.36	<code>print([*objects], {sep, end, file, flush})</code>	342
17.14.37	<code>range([start], stop, [step])</code>	343
17.14.38	<code>repr(object)</code>	344
17.14.39	<code>reversed(sequence)</code>	344
17.14.40	<code>round(x, [n])</code>	345
17.14.41	<code>set([iterable])</code>	345
17.14.42	<code>sorted(iterable, [key, reverse])</code>	345
17.14.43	<code>str([object, encoding, errors])</code>	345
17.14.44	<code>sum(iterable, [start])</code>	347
17.14.45	<code>tuple([iterable])</code>	347
17.14.46	<code>type(object)</code>	347
17.14.47	<code>zip([*iterables], {strict})</code>	348

18 Module und Pakete 349

18.1	Einbinden globaler Module	350
18.2	Lokale Module	352
18.2.1	Namenskonflikte	353
18.2.2	Modulinterne Referenzen	354
18.2.3	Module ausführen	354
18.3	Pakete	355
18.3.1	Importieren aller Module eines Pakets	357
18.3.2	Namespace Packages	358
18.3.3	Relative Importanweisungen	359
18.4	Das Paket <code>importlib</code>	359
18.4.1	Einbinden von Modulen und Paketen	360
18.4.2	Verändern des Importverhaltens	361
18.5	Geplante Sprachelemente	363

19 Objektorientierte Programmierung 365

19.1	Beispiel: Ein nicht objektorientiertes Konto	365
19.1.1	Ein neues Konto anlegen	366
19.1.2	Geld überweisen	366
19.1.3	Geld ein- und auszahlen	367
19.1.4	Den Kontostand anzeigen	368

19.2 Klassen	370
19.2.1 Definieren von Methoden	372
19.2.2 Der Konstruktor	372
19.2.3 Attribute	373
19.2.4 Beispiel: Ein objektorientiertes Konto	374
19.3 Vererbung	376
19.3.1 Ein einfaches Beispiel	376
19.3.2 Überschreiben von Methoden	377
19.3.3 Beispiel: Girokonto mit Tagesumsatz	380
19.3.4 Ausblick	388
19.4 Mehrfachvererbung	389
19.4.1 Mögliche Probleme der Mehrfachvererbung	390
19.5 Property-Attribute	390
19.5.1 Setter und Getter	390
19.5.2 Property-Attribute definieren	392
19.6 Statische Methoden	393
19.6.1 Statische Methoden definieren	393
19.7 Klassenmethoden	394
19.8 Klassenattribute	396
19.9 Built-in Functions für die objektorientierte Programmierung	396
19.9.1 Funktionen für die Verwaltung der Attribute einer Instanz	397
19.9.2 Funktionen für Informationen über die Klassenhierarchie	398
19.10 Erben von eingebauten Datentypen	400
19.11 Magic Methods und Magic Attributes	401
19.11.1 Allgemeine Magic Methods	402
19.11.2 Operatoren überladen	409
19.11.3 Datentypen emulieren – Duck-Typing	416
19.12 Datenklassen	421
19.12.1 Veränderliche Datenklassen	424
19.12.2 Unveränderliche Datenklassen	425
19.12.3 Defaultwerte in Datenklassen	425

20 Ausnahmebehandlung 427

20.1 Exceptions	427
20.1.1 Eingebaute Exceptions	428

20.1.2	Das Werfen einer Exception	429
20.1.3	Das Abfangen einer Exception	430
20.1.4	Eigene Exceptions	435
20.1.5	Erneutes Werfen einer Exception	437
20.1.6	Exception Chaining	439
20.2	Zusicherungen – assert	440
20.3	Warnungen	442

21 Generatoren und Iteratoren 445

21.1	Generatoren	445
21.1.1	Subgeneratoren	448
21.1.2	Generator Expressions	451
21.2	Iteratoren	452
21.2.1	Das Iteratorprotokoll	453
21.2.2	Beispiel: Die Fibonacci-Folge	453
21.2.3	Beispiel: Der Goldene Schnitt	455
21.2.4	Ein Generator zur Implementierung von <code>__iter__</code>	455
21.2.5	Verwendung von Iteratoren	456
21.2.6	Mehrere Iteratoren für dieselbe Instanz	459
21.2.7	Nachteile von Iteratoren gegenüber dem direkten Zugriff über Indizes	461
21.2.8	Alternative Definition für iterierbare Objekte	461
21.2.9	Funktionsiteratoren	462
21.3	Spezielle Generatoren – itertools	463
21.3.1	<code>accumulate(iterable, [func])</code>	465
21.3.2	<code>chain([*iterables])</code>	465
21.3.3	<code>combinations(iterable, r)</code>	465
21.3.4	<code>combinations_with_replacement(iterable, r)</code>	466
21.3.5	<code>compress(data, selectors)</code>	466
21.3.6	<code>count([start, step])</code>	467
21.3.7	<code>cycle(iterable)</code>	467
21.3.8	<code>dropwhile(predicate, iterable)</code>	468
21.3.9	<code>filterfalse(predicate, iterable)</code>	468
21.3.10	<code>groupby(iterable, [key])</code>	468
21.3.11	<code>islice(iterable, [start], stop, [step])</code>	469
21.3.12	<code>permutations(iterable, [r])</code>	470

21.3.13	product([*iterables], [repeat])	470
21.3.14	repeat(object, [times])	471
21.3.15	starmap(function, iterable)	471
21.3.16	takewhile(predicate, iterable)	471
21.3.17	tee(iterable, [n])	471
21.3.18	zip_longest([*iterables], {fillvalue})	472

22 Kontext-Manager 473

22.1	Die with-Anweisung	473
22.2	Hilfsfunktionen für with-Kontexte – contextlib	476
22.2.1	Dynamisch zusammengestellte Kontext-Kombinationen – ExitStack	476
22.2.2	Bestimmte Exception-Typen unterdrücken	477
22.2.3	Den Standard-Ausgabestrom umleiten	478
22.2.4	Optionale Kontexte	479
22.2.5	Einfache Funktionen als Kontext-Manager	479

23 Dekoratoren 481

23.1	Funktionsdekoratoren	481
23.1.1	Das Dekorieren von Funktionen und Methoden	483
23.1.2	Name und Docstring nach Anwendung eines Dekorators	483
23.1.3	Verschachtelte Dekoratoren	484
23.1.4	Beispiel: Ein Cache-Dekorator	485
23.2	Klassendekoratoren	487
23.3	Das Modul functools	488
23.3.1	Funktionsschnittstellen vereinfachen	488
23.3.2	Methodenschnittstellen vereinfachen	489
23.3.3	Caches	490
23.3.4	Ordnungsrelationen vervollständigen	492
23.3.5	Überladen von Funktionen	492

24 Annotationen und statische Typprüfung 495

24.1 Annotationen	497
24.1.1 Die Annotation von Funktionen und Methoden	498
24.1.2 Die Annotation von Variablen und Attributen	499
24.1.3 Der Zugriff auf Annotationen zur Laufzeit	501
24.1.4 Wann werden Annotationen evaluiert?	503
24.2 Type Hints – das Modul typing	504
24.2.1 Gültige Type Hints	505
24.2.2 Containertypen	505
24.2.3 Abstrakte Containertypen	506
24.2.4 Typ-Alias	507
24.2.5 Type Unions und optionale Werte	507
24.2.6 Typvariablen	508
24.3 Statische Typprüfung in Python – mypy	509
24.3.1 Installation	510
24.3.2 Beispiel	510

25 Structural Pattern Matching 513

25.1 Die match-Anweisung	513
25.2 Arten von Mustern in der case-Anweisung	514
25.2.1 Literal- und Wertmuster	515
25.2.2 Oder-Muster	515
25.2.3 Muster mit Typprüfung	516
25.2.4 Bedingungen für Matches formulieren	517
25.2.5 Teilmuster gruppieren	518
25.2.6 Capture- und Wildcard-Muster	518
25.2.7 Sequenzmuster	520
25.2.8 Zuordnungsmuster	522
25.2.9 Muster für Objekte und ihre Attributwerte	525

TEIL IV Die Standardbibliothek

26 Mathematik	533
26.1 Mathematische Funktionen – math, cmath	533
26.1.1 Allgemeine mathematische Funktionen	534
26.1.2 Exponential- und Logarithmusfunktionen	537
26.1.3 Trigonometrische und hyperbolische Funktionen	537
26.1.4 Distanzen und Normen	538
26.1.5 Umrechnen von Winkeln	538
26.1.6 Darstellungsformen komplexer Zahlen	539
26.2 Zufallszahlengenerator – random	539
26.2.1 Den Status des Zufallszahlengenerators speichern und laden	540
26.2.2 Zufällige ganze Zahlen erzeugen	541
26.2.3 Zufällige Gleitkommazahlen erzeugen	541
26.2.4 Zufallsgesteuerte Operationen auf Sequenzen	542
26.2.5 SystemRandom([seed])	543
26.3 Statistische Berechnungen – statistics	544
26.4 Intuitive Dezimalzahlen – decimal	545
26.4.1 Verwendung des Datentyps	546
26.4.2 Nichtnumerische Werte	549
26.4.3 Das Context-Objekt	550
26.5 Hash-Funktionen – hashlib	551
26.5.1 Verwendung des Moduls	553
26.5.2 Weitere Hash-Algorithmen	555
26.5.3 Vergleich großer Dateien	555
26.5.4 Passwörter	557
27 Bildschirmausgaben und Logging	559
27.1 Übersichtliche Ausgabe komplexer Objekte – pprint	559
27.2 Logdateien – logging	561
27.2.1 Das Meldungsformat anpassen	563
27.2.2 Logging-Handler	565

28 Reguläre Ausdrücke 569

28.1 Die Syntax regulärer Ausdrücke	569
28.1.1 Beliebige Zeichen	570
28.1.2 Zeichenklassen	570
28.1.3 Quantoren	571
28.1.4 Vordefinierte Zeichenklassen	573
28.1.5 Weitere Sonderzeichen	575
28.1.6 Genügsame Quantoren	576
28.1.7 Gruppen	577
28.1.8 Alternativen	578
28.1.9 Extensions	578
28.2 Verwendung des Moduls re	581
28.2.1 Searching	581
28.2.2 Matching	582
28.2.3 Einen String aufspalten	583
28.2.4 Teile eines Strings ersetzen	583
28.2.5 Problematische Zeichen ersetzen	584
28.2.6 Einen regulären Ausdruck kompilieren	585
28.2.7 Flags	585
28.2.8 Das Match-Objekt	587
28.3 Ein einfaches Beispielprogramm – Searching	588
28.4 Ein komplexeres Beispielprogramm – Matching	590
28.5 Kommentare in regulären Ausdrücken	593

29 Schnittstellen zum Betriebssystem und zur Laufzeitumgebung 595

29.1 Funktionen des Betriebssystems – os	595
29.1.1 environ	596
29.1.2 getpid()	596
29.1.3 cpu_count()	596
29.1.4 system(cmd)	596
29.1.5 popen(command, [mode, buffering])	597
29.2 Zugriff auf die Laufzeitumgebung – sys	597
29.2.1 Kommandozeilenparameter	598
29.2.2 Standardpfade	598

29.2.3	Standard-Ein-/Ausgabeströme	598
29.2.4	Das Programm beenden	599
29.2.5	Details zur Python-Version	599
29.2.6	Details zum Betriebssystem	600
29.2.7	Hooks	601
29.3	Kommandozeilenparameter – argparse	602
29.3.1	Taschenrechner – ein einfaches Beispiel	604
29.3.2	Ein komplexeres Beispiel	608

30	Das Dateisystem	611
-----------	------------------------------	------------

30.1	Zugriff auf das Dateisystem mit os	611
30.2	Dateipfade – os.path	618
30.3	Zugriff auf das Dateisystem – shutil	622
30.3.1	Verzeichnis- und Dateioperationen	624
30.3.2	Archivoperationen	626
30.4	Temporäre Dateien – tempfile	628

31	Parallele Programmierung	631
-----------	---------------------------------------	------------

31.1	Prozesse, Multitasking und Threads	631
31.1.1	Die Leichtgewichte unter den Prozessen – Threads	632
31.1.2	Threads oder Prozesse?	634
31.1.3	Kooperatives Multitasking – ein dritter Weg	634
31.2	Pythons Schnittstellen zur Parallelisierung	635
31.3	Die abstrakte Schnittstelle – concurrent.futures	636
31.3.1	Ein Beispiel mit einem futures.ThreadPoolExecutor	637
31.3.2	Executor-Instanzen als Kontext-Manager	639
31.3.3	Die Verwendung von futures.ProcessPoolExecutor	640
31.3.4	Die Verwaltung der Aufgaben eines Executors	641
31.4	Die flexible Schnittstelle – threading und multiprocessing	648
31.4.1	Threads in Python – threading	648
31.4.2	Prozesse in Python – multiprocessing	658
31.5	Die kooperative Schnittstelle – asyncio	660
31.5.1	Kooperative Funktionen – Koroutinen	661

31.5.2	Erwartbare Objekte	661
31.5.3	Die Kooperation von Koroutinen – Tasks	662
31.5.4	Ein kooperativer Webcrawler	665
31.5.5	Blockierende Operationen in Koroutinen	674
31.5.6	Weitere asynchrone Sprachmerkmale	676
31.6	Fazit: Welche Schnittstelle ist die richtige?	678

32 Datenspeicherung 681

32.1	XML	681
32.1.1	ElementTree	683
32.1.2	SAX – Simple API for XML	691
32.2	Datenbanken	695
32.2.1	Pythons eingebaute Datenbank – sqlite3	698
32.3	Komprimierte Dateien und Archive	715
32.3.1	gzip.open(filename, [mode, compresslevel])	715
32.3.2	Andere Module für den Zugriff auf komprimierte Daten	716
32.4	Serialisierung von Instanzen – pickle	716
32.4.1	Funktionale Schnittstelle	718
32.4.2	Objektorientierte Schnittstelle	719
32.5	Das Datenaustauschformat JSON – json	720
32.6	Das Tabellenformat CSV – csv	722
32.6.1	reader-Objekte – Daten aus einer CSV-Datei lesen	722
32.6.2	Dialect-Objekte – eigene Dialekte verwenden	725

33 Netzwerkkommunikation 729

33.1	Die Socket API	730
33.1.1	Client-Server-Systeme	731
33.1.2	UDP	734
33.1.3	TCP	735
33.1.4	Blockierende und nichtblockierende Sockets	737
33.1.5	Erzeugen eines Sockets	739
33.1.6	Die Socket-Klasse	740
33.1.7	Netzwerk-Byte-Order	743

33.1.8	Multiplexende Server – selectors	744
33.1.9	Objektorientierte Serverentwicklung – socketserver	747
33.2	XML-RPC	749
33.2.1	Der Server	750
33.2.2	Der Client	753
33.2.3	Multicall	755
33.2.4	Einschränkungen	756

34 Zugriff auf Ressourcen im Internet 759

34.1	Protokolle	759
34.1.1	Hypertext Transfer Protocol – HTTP	759
34.1.2	File Transfer Protocol – FTP	760
34.2	Lösungen	760
34.2.1	Veraltete Lösungen für Python 2	760
34.2.2	Lösungen der Standardbibliothek	760
34.2.3	Lösungen von Drittanbietern	760
34.3	Der einfache Weg – requests	761
34.3.1	Einfache Anfragen via GET und POST	761
34.3.2	Web-APIs	762
34.4	URLs – urllib	764
34.4.1	Zugriff auf entfernte Ressourcen – urllib.request	764
34.4.2	Das Einlesen und Verarbeiten von URLs – urllib.parse	768
34.5	FTP – ftplib	772
34.5.1	Mit einem FTP-Server verbinden	774
34.5.2	FTP-Kommandos ausführen	775
34.5.3	Mit Dateien und Verzeichnissen arbeiten	775
34.5.4	Übertragen von Dateien	776

35 E-Mail 781

35.1	SMTP – smtplib	781
35.1.1	SMTP([host, port, local_hostname, timeout, source_address])	782
35.1.2	Eine Verbindung aufbauen und beenden	783
35.1.3	Eine E-Mail versenden	783
35.1.4	Beispiel	784

35.2 POP3 – poplib	784
35.2.1 POP3(host, [port, timeout])	785
35.2.2 Eine Verbindung aufbauen und beenden	786
35.2.3 Vorhandene E-Mails auflisten	786
35.2.4 E-Mails abrufen und löschen	787
35.2.5 Beispiel	788
35.3 IMAP4 – imaplib	789
35.3.1 IMAP4([host, port, timeout])	790
35.3.2 Eine Verbindung aufbauen und beenden	790
35.3.3 Eine Mailbox suchen und auswählen	791
35.3.4 Operationen mit Mailboxen	792
35.3.5 E-Mails suchen	792
35.3.6 E-Mails abrufen	793
35.3.7 Beispiel	794
35.4 Erstellen komplexer E-Mails – email	795
35.4.1 Eine einfache E-Mail erstellen	795
35.4.2 Eine E-Mail mit Anhängen erstellen	796
35.4.3 Eine E-Mail einlesen	798

36 Debugging und Qualitätssicherung 799

36.1 Der Debugger	799
36.2 Automatisiertes Testen	802
36.2.1 Testfälle in Docstrings – doctest	802
36.2.2 Unit Tests – unittest	807
36.3 Analyse des Laufzeitverhaltens	810
36.3.1 Laufzeitmessung – timeit	811
36.3.2 Profiling – cProfile	814
36.3.3 Tracing – trace	818

37 Dokumentation 821

37.1 Docstrings	821
37.2 Automatisches Erstellen einer Dokumentation – pydoc	823

TEIL V Weiterführende Themen

38	Distribution von Python-Projekten	827
<hr/>		
38.1	Eine Geschichte der Distributionen in Python	827
38.1.1	Der klassische Ansatz – distutils	828
38.1.2	Der neue Standard – setuptools	828
38.1.3	Der Paketindex – PyPI	829
38.2	Erstellen von Distributionen – setuptools	829
38.2.1	Installation	829
38.2.2	Schreiben des Moduls	830
38.2.3	Das Installationskript	831
38.2.4	Erstellen einer Quellcodedistribution	836
38.2.5	Erstellen einer Binärdistribution	836
38.2.6	Distributionen installieren	837
38.3	Erstellen von EXE-Dateien – cx_Freeze	838
38.4	Paketmanager	840
38.4.1	Der Python-Paketmanager – pip	840
38.4.2	Der Paketmanager conda	842
38.5	Lokalisierung von Programmen – gettext	845
38.5.1	Beispiel für die Verwendung von gettext	846
38.5.2	Erstellen des Sprachkompilats	847
39	Virtuelle Umgebungen	851
<hr/>		
39.1	Das Arbeiten mit virtuellen Umgebungen – venv	852
39.1.1	Eine virtuelle Umgebung aktivieren	852
39.1.2	In einer virtuellen Umgebung arbeiten	852
39.1.3	Eine virtuelle Umgebung deaktivieren	853
39.2	Virtuelle Umgebungen in Anaconda	853
40	Alternative Interpreter und Compiler	855
<hr/>		
40.1	Just-in-Time-Kompilierung – PyPy	855
40.1.1	Installation und Verwendung	856
40.1.2	Beispiel	856

40.2 Numba	856
40.2.1 Installation	857
40.2.2 Beispiel	858
40.3 Anbindung an C und C++ – Cython	859
40.3.1 Installation	859
40.3.2 Die Funktionsweise von Cython	860
40.3.3 Ein Cython-Programm kompilieren	861
40.3.4 Ein Cython-Programm mit statischer Typisierung	862
40.3.5 Eine C-Bibliothek verwenden	864
40.4 Die interaktive Python-Shell – IPython	866
40.4.1 Installation	866
40.4.2 Die interaktive Shell	867
40.4.3 Das Jupyter Notebook	869

41 Grafische Benutzeroberflächen 873

41.1 Toolkits	873
41.1.1 Tkinter (Tk)	874
41.1.2 PyGObject (Gtk)	874
41.1.3 Qt for Python (Qt)	874
41.1.4 wxPython (wxWidgets)	875
41.2 Einführung in tkinter	875
41.2.1 Ein einfaches Beispiel	876
41.2.2 Steuerelementvariablen	878
41.2.3 Der Packer	880
41.2.4 Events	884
41.2.5 Steuerelemente	891
41.2.6 Zeichnungen – das Canvas-Widget	909
41.2.7 Weitere Module	917
41.3 Einführung in PySide6	920
41.3.1 Installation	921
41.3.2 Grundlegende Konzepte von Qt	921
41.3.3 Der Entwicklungsprozess	923
41.4 Signale und Slots	930
41.5 Wichtige Widgets	933
41.5.1 QCheckBox	933
41.5.2 QComboBox	934

- 41.5.3 QDateEdit, QtimeEdit und QDateTimeEdit 934
 - 41.5.4 QDialog 935
 - 41.5.5 QLineEdit 936
 - 41.5.6 QListWidget und QListView 936
 - 41.5.7 QProgressBar 937
 - 41.5.8 QPushButton 937
 - 41.5.9 QRadioButton 938
 - 41.5.10 QSlider und QDial 938
 - 41.5.11 QTextEdit 939
 - 41.5.12 QWidget 939
- 41.6 Die Zeichenfunktionalität von Qt 940**
 - 41.6.1 Werkzeuge 941
 - 41.6.2 Das Koordinatensystem 943
 - 41.6.3 Einfache Formen 944
 - 41.6.4 Grafiken 946
 - 41.6.5 Text 947
 - 41.6.6 Eye Candy 949
- 41.7 Die Model-View-Architektur 953**
 - 41.7.1 Beispielprojekt: ein Adressbuch 954
 - 41.7.2 Auswählen von Einträgen 963
 - 41.7.3 Bearbeiten von Einträgen 964

42 Python als serverseitige Programmiersprache im WWW – ein Einstieg in Django

969

- 42.1 Konzepte und Besonderheiten von Django 970**
- 42.2 Installation von Django 971**
- 42.3 Ein neues Django-Projekt erstellen 972**
 - 42.3.1 Der Entwicklungsserver 973
 - 42.3.2 Konfiguration des Projekts 975
- 42.4 Eine Applikation erstellen 976**
 - 42.4.1 Die Applikation in das Projekt einbinden 978
 - 42.4.2 Ein Model definieren 978
 - 42.4.3 Beziehungen zwischen Modellen 979
 - 42.4.4 Übertragung des Modells in die Datenbank 980
 - 42.4.5 Die Model-API 981
 - 42.4.6 Unser Projekt bekommt ein Gesicht 987
 - 42.4.7 Djangos Template-System 994

42.4.8	Verarbeitung von Formulardaten	1006
42.4.9	Djangos Administrationsoberfläche	1010

43 Wissenschaftliches Rechnen und Data Science 1017

43.1	Installation	1018
43.2	Das Modellprogramm	1018
43.2.1	Der Import von numpy, scipy und matplotlib	1020
43.2.2	Vektorisierung und der Datentyp numpy.ndarray	1020
43.2.3	Visualisieren von Daten mit matplotlib.pyplot	1024
43.3	Überblick über die Module numpy und scipy	1027
43.3.1	Überblick über den Datentyp numpy.ndarray	1027
43.3.2	Überblick über scipy	1036
43.4	Eine Einführung in die Datenanalyse mit pandas	1038
43.4.1	Das DataFrame-Objekt	1039
43.4.2	Selektiver Datenzugriff	1040
43.4.3	Löschen von Zeilen und Spalten	1046
43.4.4	Einfügen von Zeilen und Spalten	1046
43.4.5	Logische Ausdrücke auf Datensätzen	1047
43.4.6	Manipulation von Datensätzen	1048
43.4.7	Ein- und Ausgabe	1050
43.4.8	Visualisierung	1051

44 Insiderwissen 1055

44.1	URLs im Standardbrowser öffnen – webbrowser	1055
44.2	Interpretieren von Binärdaten – struct	1055
44.3	Versteckte Passworteingabe – getpass	1058
44.4	Kommandozeilen-Interpreter – cmd	1058
44.5	Dateiinterface für Strings – io.StringIO	1061
44.6	Generatoren als Konsumenten	1062
44.6.1	Ein Decorator für konsumierende Generatorfunktionen	1064
44.6.2	Auslösen von Exceptions in einem Generator	1065
44.6.3	Eine Pipeline als Verkettung konsumierender Generator- funktionen	1066

44.7 Kopieren von Instanzen – copy	1068
44.8 Bildverarbeitung – Pillow	1071
44.8.1 Installation	1071
44.8.2 Bilddateien laden und speichern	1072
44.8.3 Zugriff auf einzelne Pixel	1073
44.8.4 Manipulation von Bildern	1073
44.8.5 Interoperabilität	1080

45 Von Python 2 nach Python 3 **1081**

45.1 Die wichtigsten Unterschiede	1084
45.1.1 Ein-/Ausgabe	1085
45.1.2 Iteratoren	1086
45.1.3 Strings	1086
45.1.4 Ganze Zahlen	1088
45.1.5 Exception Handling	1088
45.1.6 Standardbibliothek	1089
45.2 Automatische Konvertierung	1090

A Anhang **1095**

A.1 Reservierte Wörter	1095
A.2 Operatorrangfolge	1095
A.3 Eingebaute Funktionen	1097
A.4 Eingebaute Exceptions	1101
A.5 Python-IDEs	1105

Index	1109
--------------------	-------------