

Johannes Ernesti, Peter Kaiser

Python 3

Das umfassende Handbuch

Inhalt

1	Einleitung	25
<hr/>		
2	Die Programmiersprache Python	31
<hr/>		
2.1	Historie, Konzepte, Einsatzgebiete	31
2.1.1	Geschichte und Entstehung	31
2.1.2	Grundlegende Konzepte	32
2.1.3	Einsatzmöglichkeiten und Stärken	33
2.1.4	Einsatzbeispiele	34
2.2	Die Verwendung von Python	34
2.2.1	Windows	36
2.2.2	Linux	36
2.2.3	OS X	36

TEIL I Einstieg in Python

3	Erste Schritte im interaktiven Modus	39
<hr/>		
3.1	Ganze Zahlen	40
3.2	Gleitkommazahlen	41
3.3	Zeichenketten	42
3.4	Listen	42
3.5	Dictionarys	43
3.6	Variablen	44
3.7	Logische Ausdrücke	46
3.8	Funktionen und Methoden	47
3.8.1	Funktionen	48
3.8.2	Methoden	48
3.9	Bildschirm Ausgaben	49

4	Der Weg zum ersten Programm	53
<hr/>		
4.1	Tippen, kompilieren, testen	53
4.1.1	Shebang	55
4.1.2	Interne Abläufe	55
4.2	Grundstruktur eines Python-Programms	57
4.2.1	Umbrechen langer Zeilen	59
4.2.2	Zusammenfügen mehrerer Zeilen	59
4.3	Das erste Programm	60
4.4	Kommentare	63
4.5	Der Fehlerfall	63
5	Kontrollstrukturen	65
<hr/>		
5.1	Fallunterscheidungen	65
5.1.1	Die if-Anweisung	65
5.1.2	Bedingte Ausdrücke	69
5.2	Schleifen	70
5.2.1	Die while-Schleife	70
5.2.2	Abbruch einer Schleife	71
5.2.3	Erkennen eines Schleifenabbruchs	72
5.2.4	Abbruch eines Schleifendurchlaufs	73
5.2.5	Die for-Schleife	75
5.2.6	Die for-Schleife als Zählschleife	77
5.3	Die pass-Anweisung	78
6	Dateien	79
<hr/>		
6.1	Datenströme	79
6.2	Daten aus einer Datei auslesen	80
6.3	Daten in eine Datei schreiben	84
6.4	Das Dateiojekt erzeugen	85
6.4.1	open(filename, [mode, buffering, encoding, errors, newline])	85
6.4.2	Attribute und Methoden eines Dateiojekts	87
6.4.3	Die Schreib-/Leseposition verändern	88

7 Das Laufzeitmodell 91

7.1 Die Struktur von Instanzen	93
7.1.1 Datentyp	93
7.1.2 Wert	94
7.1.3 Identität	95
7.2 Referenzen und Instanzen freigeben	97
7.3 Mutable vs. immutable Datentypen	98
7.3.1 Mutable Datentypen und Seiteneffekte	100

8 Funktionen, Methoden und Attribute 103

8.1 Parameter von Funktionen und Methoden	103
8.1.1 Positionsbezogene Parameter	104
8.1.2 Schlüsselwortparameter	105
8.1.3 Optionale Parameter	105
8.1.4 Reine Schlüsselwortparameter	106
8.2 Attribute	106

9 Informationsquellen zu Python 109

9.1 Die Built-in Funktion help	109
9.2 Die Onlinedokumentation	110
9.3 PEPs	110

TEIL II Datentypen

10 Das Nichts – NoneType 115

11 Operatoren 117

12 Numerische Datentypens 121

12.1	Arithmetische Operatoren	121
12.2	Vergleichende Operatoren	123
12.3	Konvertierung zwischen numerischen Datentypen	124
12.4	Ganzzahlen – int	125
	12.4.1 Zahlensysteme	125
	12.4.2 Bit-Operationen	127
	12.4.3 Methoden	130
12.5	Gleitkommazahlen – float	130
12.6	Boolesche Werte – bool	133
	12.6.1 Logische Operatoren	133
	12.6.2 Wahrheitswerte nicht-boolescher Datentypen	136
	12.6.3 Auswertung logischer Operatoren	137
12.7	Komplexe Zahlen – complex	138

13 Sequenzielle Datentypen 141

13.1	Operationen auf Instanzen sequenzieller Datentypen	142
	13.1.1 Ist ein Element vorhanden? – die Operatoren in und not in	143
	13.1.2 Verkettung von Sequenzen – die Operatoren + und +=	145
	13.1.3 Wiederholung von Sequenzen – die Operatoren * und *=	146
	13.1.4 Zugriff auf bestimmte Elemente einer Sequenz – der []-Operator	147
	13.1.5 Länge einer Sequenz – die Built-in Funktion len	151
	13.1.6 Das kleinste und das größte Element einer Sequenz – min und max	152
	13.1.7 Die Position eines Elements in der Sequenz – s.index(x, [i, j])	152
	13.1.8 Anzahl der Vorkommen eines Elements der Sequenz – s.count(x)	153
13.2	Listen – list	154
	13.2.1 Verändern eines Wertes innerhalb der Liste – Zuweisung mit []	155
	13.2.2 Ersetzen von Teillisten und Einfügen neuer Elemente – Zuweisung mit []	155
	13.2.3 Elemente und Teillisten löschen – del zusammen mit []	156
	13.2.4 Methoden von list-Instanzen	156
	13.2.5 Weitere Eigenschaften von Listen	163

13.3 Unveränderliche Listen – tuple	166
13.3.1 Tuple Packing/Unpacking und Sequence Unpacking	166
13.3.2 Immutabel heißt nicht zwingend unveränderlich!	168
13.4 Strings – str, bytes, bytearray	168
13.4.1 Steuerzeichen	171
13.4.2 String-Methoden	173
13.4.3 Formatierung von Strings	183
13.4.4 Zeichensätze und Sonderzeichen	192

14 Zuordnungen 201

14.1 Dictionary – dict	201
14.1.1 Operatoren	204
14.1.2 Methoden	206

15 Mengen 213

15.1 Die Datentypen set und frozenset	213
15.1.1 Operatoren	214
15.1.2 Methoden	220
15.2 Veränderliche Mengen – set	221
15.3 Unveränderliche Mengen – frozenset	223

16 Collections 225

16.1 Verkettete Dictionarys	225
16.2 Zählen von Häufigkeiten	226
16.3 Dictionarys mit Standardwerten	229
16.4 Doppelt verkettete Listen	230
16.5 Benannte Tupel	232
16.6 Sortierte Dictionarys	233

17 Datum und Zeit 235

17.1 Elementare Zeitfunktionen – time	235
17.1.1 Attribute	237
17.1.2 Funktionen	238
17.2 Objektorientierte Datumsverwaltung – datetime	243
17.2.1 datetime.date	244
17.2.2 datetime.time	245
17.2.3 datetime.datetime	246
17.2.4 datetime.timedelta	248
17.2.5 Operationen für datetime.datetime und datetime.date	251
17.2.6 Bemerkung zum Umgang mit Zeitzonen	253

18 Aufzählungstypen – Enum 255

TEIL III Fortgeschrittene Programmiertechniken

19 Funktionen 261

19.1 Schreiben einer Funktion	263
19.2 Funktionsparameter	267
19.2.1 Optionale Parameter	267
19.2.2 Schlüsselwortparameter	268
19.2.3 Beliebige Anzahl von Parametern	269
19.2.4 Reine Schlüsselwortparameter	271
19.2.5 Entpacken einer Parameterliste	272
19.2.6 Seiteneffekte	274
19.3 Namensräume	277
19.3.1 Zugriff auf globale Variablen – global	277
19.3.2 Zugriff auf den globalen Namensraum	278
19.3.3 Lokale Funktionen	279
19.3.4 Zugriff auf übergeordnete Namensräume – nonlocal	280
19.4 Anonyme Funktionen	281
19.5 Annotationen	282
19.6 Rekursion	284

19.7 Eingebaute Funktionen	285
19.7.1 abs(x)	288
19.7.2 all(iterable)	289
19.7.3 any(iterable)	289
19.7.4 ascii(object)	289
19.7.5 bin(x)	290
19.7.6 bool([x])	290
19.7.7 bytearray([source, encoding, errors])	290
19.7.8 bytes([source, encoding, errors])	291
19.7.9 chr(i)	291
19.7.10 complex([real, imag])	292
19.7.11 dict([source])	292
19.7.12 divmod(a, b)	293
19.7.13 enumerate(iterable)	293
19.7.14 eval(expression, [globals, locals])	294
19.7.15 exec(object, [globals, locals])	294
19.7.16 filter(function, iterable)	295
19.7.17 float([x])	295
19.7.18 format(value, [format_spec])	296
19.7.19 frozenset([iterable])	296
19.7.20 globals()	296
19.7.21 hash(object)	297
19.7.22 help([object])	298
19.7.23 hex(x)	298
19.7.24 id(object)	298
19.7.25 input([prompt])	298
19.7.26 int([x, base])	299
19.7.27 len(s)	299
19.7.28 list([sequence])	300
19.7.29 locals()	300
19.7.30 map(function, [*iterable])	300
19.7.31 max(iterable, {default, key}) max(arg1, arg2, [*args], {key})	302
19.7.32 min(iterable, {default, key}) min(arg1, arg2, [*args], {key})	303
19.7.33 oct(x)	303
19.7.34 ord(c)	303
19.7.35 pow(x, y, [z])	303
19.7.36 print([*objects], {sep, end, file, flush})	304
19.7.37 range([start], stop, [step])	304
19.7.38 repr(object)	305

19.7.39	reversed(sequence)	306
19.7.40	round(x, [n])	306
19.7.41	set([iterable])	306
19.7.42	sorted(iterable, [key, reverse])	307
19.7.43	str([object, encoding, errors])	307
19.7.44	sum(iterable, [start])	308
19.7.45	tuple([iterable])	309
19.7.46	type(object)	309
19.7.47	zip([*iterables])	309

20 Modularisierung 311

20.1	Einbinden globaler Module	311
20.2	Lokale Module	314
20.2.1	Namenskonflikte	315
20.2.2	Modulinterne Referenzen	316
20.2.3	Module ausführen	316
20.3	Pakete	317
20.3.1	Importieren aller Module eines Pakets	319
20.3.2	Namespace Packages	320
20.3.3	Relative Import-Anweisungen	320
20.4	Das Paket importlib	321
20.4.1	Einbinden von Modulen und Paketen	322
20.4.2	Verändern des Import-Verhaltens	322

21 Objektorientierung 327

21.1	Klassen	332
21.1.1	Definieren von Methoden	333
21.1.2	Der Konstruktor und die Erzeugung von Attributen	334
21.2	Vererbung	337
21.2.1	Technische Grundlagen	338
21.2.2	Die Klasse GirokontoMitTagesumsatz	341
21.2.3	Mögliche Erweiterungen der Klasse Konto	346
21.2.4	Ausblick	350
21.2.5	Mehrfachvererbung	351

21.3	Setter und Getter und Property Attributes	352
21.3.1	Setter und Getter	352
21.3.2	Property-Attribute	353
21.4	Klassenattribute und Klassenmethoden sowie statische Methoden	355
21.4.1	Statische Methoden	355
21.4.2	Klassenmethoden	356
21.4.3	Klassenattribute	357
21.5	Built-in Functions für Objektorientierung	358
21.5.1	Funktionen für die Verwaltung der Attribute einer Instanz	359
21.5.2	Funktionen für Informationen über die Klassenhierarchie	360
21.6	Objektphilosophie	361
21.7	Magic Methods und Magic Attributes	363
21.7.1	Allgemeine Magic Methods	364
21.7.2	Operatoren überladen	370
21.7.3	Datentypen emulieren	378

22 Ausnahmebehandlung 383

22.1	Exceptions	383
22.1.1	Eingebaute Exceptions	384
22.1.2	Werfen einer Exception	385
22.1.3	Abfangen einer Exception	386
22.1.4	Eigene Exceptions	390
22.1.5	Erneutes Werfen einer Exception	392
22.1.6	Exception Chaining	395
22.2	Zusicherungen – assert	396

23 Iteratoren und Generatoren 399

23.1	Comprehensions	399
23.1.1	List Comprehensions	399
23.1.2	Dict Comprehensions	402
23.1.3	Set Comprehensions	403
23.2	Generatoren	403
23.2.1	Subgeneratoren	406
23.2.2	Generator Expressions	409

23.3 Iteratoren	410
23.3.1 Verwendung von Iteratoren	413
23.3.2 Mehrere Iteratoren für dieselbe Instanz	416
23.3.3 Nachteile von Iteratoren gegenüber dem direkten Zugriff über Indizes	419
23.3.4 Alternative Definition für iterierbare Objekte	419
23.3.5 Funktionsiteratoren	420
23.4 Spezielle Generatoren – itertools	421

24 Kontextobjekte 431

24.1 Die with-Anweisung	431
24.2 Hilfsfunktionen für with-Kontexte – contextlib	434
24.2.1 Einfache Funktionen als Kontext-Manager	434
24.2.2 Bestimmte Exception-Typen unterdrücken	435
24.2.3 Den Standard-Ausgabestrom umleiten	436

25 Manipulation von Funktionen und Methoden 437

25.1 Decorator	437
25.2 Das Modul functools	440
25.2.1 Funktionsschnittstellen vereinfachen	440
25.2.2 Methodenschnittstellen vereinfachen	442
25.2.3 Caches	443
25.2.4 Ordnungsrelationen vervollständigen	444
25.2.5 Überladen von Funktionen	445

TEIL IV Die Standardbibliothek

26 Mathematik 451

26.1 Mathematische Funktionen – math, cmath	451
26.1.1 Zahlentheoretische Funktionen	452
26.1.2 Exponential- und Logarithmusfunktionen	454
26.1.3 Trigonometrische und hyperbolische Funktionen	454

26.1.4	Umrechnen von Winkeln	455
26.1.5	Darstellungsformen komplexer Zahlen	455
26.2	Zufallszahlengenerator – random	456
26.2.1	Den Status speichern und laden	457
26.2.2	Zufällige ganze Zahlen erzeugen	457
26.2.3	Zufällige Gleitkommazahlen erzeugen	458
26.2.4	Zufallsgesteuerte Operationen auf Sequenzen	458
26.2.5	SystemRandom([seed])	459
26.3	Präzise Dezimalzahlen – decimal	460
26.3.1	Verwendung des Datentyps	461
26.3.2	Nichtnumerische Werte	464
26.3.3	Das Context-Objekt	465

27 Kryptografie 467

27.1	Hash-Funktionen – hashlib	467
27.1.1	Verwendung des Moduls	469
27.1.2	Weitere Algorithmen	470
27.1.3	Vergleich großer Dateien	470
27.1.4	Passwörter	471
27.2	Verschlüsselung – PyCrypto	472
27.2.1	Symmetrische Verschlüsselungsverfahren	473
27.2.2	Asymmetrische Verschlüsselungsverfahren	476

28 Reguläre Ausdrücke 481

28.1	Syntax regulärer Ausdrücke	481
28.1.1	Beliebige Zeichen	482
28.1.2	Zeichenklassen	482
28.1.3	Quantoren	483
28.1.4	Vordefinierte Zeichenklassen	485
28.1.5	Weitere Sonderzeichen	487
28.1.6	Genügsame Quantoren	488
28.1.7	Gruppen	489
28.1.8	Alternativen	490
28.1.9	Extensions	490

28.2	Verwendung des Moduls	493
28.2.1	Searching	493
28.2.2	Matching	494
28.2.3	Einen String aufspalten	494
28.2.4	Teile eines Strings ersetzen	495
28.2.5	Problematische Zeichen ersetzen	496
28.2.6	Einen regulären Ausdruck kompilieren	496
28.2.7	Flags	496
28.2.8	Das Match-Objekt	498
28.3	Ein einfaches Beispielprogramm – Searching	499
28.4	Ein komplexeres Beispielprogramm – Matching	500

29 Schnittstelle zu Betriebssystem und Laufzeitumgebung 505

29.1	Funktionen des Betriebssystems – os	505
29.1.1	environ	506
29.1.2	getpid()	506
29.1.3	cpu_count()	506
29.1.4	system(cmd)	507
29.1.5	popen(command, [mode, buffering])	507
29.2	Zugriff auf die Laufzeitumgebung – sys	508
29.2.1	Kommandozeilenparameter	508
29.2.2	Standardpfade	508
29.2.3	Standard-Ein-/Ausgabeströme	509
29.2.4	Das Programm beenden	509
29.2.5	Details zur Python-Version	510
29.2.6	Details zum Betriebssystem	511
29.2.7	Hooks	512

30 Kommandozeilenparameter 515

30.1	Taschenrechner – ein einfaches Beispiel	516
30.2	Ein weiteres Beispiel	520

31	Dateisystem	523
<hr/>		
31.1	Zugriff auf das Dateisystem mit <code>os</code>	523
31.2	Dateipfade – <code>os.path</code>	530
31.3	Zugriff auf das Dateisystem – <code>shutil</code>	535
31.3.1	Verzeichnis- und Dateioperationen	537
31.3.2	Archivoperationen	538
31.4	Temporäre Dateien – <code>tempfile</code>	541
32	Parallele Programmierung	543
<hr/>		
32.1	Prozesse, Multitasking und Threads	543
32.1.1	Die Leichtgewichte unter den Prozessen – Threads	544
32.1.2	Threads oder Prozesse?	546
32.2	Pythons Schnittstellen zur Parallelisierung	546
32.3	Parallelisierung von Funktionsaufrufen	547
32.3.1	Ein Beispiel mit einem <code>futures.ThreadPoolExecutor</code>	548
32.3.2	Executor-Instanzen als Kontext-Manager	550
32.3.3	Die Verwendung von <code>futures.ProcessPoolExecutor</code>	550
32.3.4	Die Verwaltung der Aufgaben eines Executors	551
32.4	Die Module <code>threading</code> und <code>multiprocessing</code>	558
32.5	Die Thread-Unterstützung in Python	558
32.5.1	Kritische Bereiche mit Lock-Objekten absichern	560
32.5.2	Datenaustausch zwischen Threads mit Critical Sections	562
32.5.3	Gefahren von Critical Sections – Deadlocks	567
32.6	Einblick in das Modul <code>multiprocessing</code>	568
32.7	Ausblick	569
33	Datenspeicherung	571
<hr/>		
33.1	Komprimierte Dateien lesen und schreiben – <code>gzip</code>	571
33.2	XML	573
33.2.1	<code>ElementTree</code>	575
33.2.2	SAX – Simple API for XML	583

33.3	Datenbanken	587
33.3.1	Pythons eingebaute Datenbank – sqlite3	590
33.4	Serialisierung von Instanzen – pickle	607
33.4.1	Funktionale Schnittstelle	608
33.4.2	Objektorientierte Schnittstelle	609
33.5	Das Datenaustauschformat JSON – json	610
33.6	Das Tabellenformat CSV – csv	612
33.6.1	reader-Objekte – Daten aus einer CSV-Datei lesen	613
33.6.2	Dialect-Objekte – eigene Dialekte verwenden	615
34	Netzwerkkommunikation	619
<hr/>		
34.1	Socket API	620
34.1.1	Client-Server-Systeme	621
34.1.2	UDP	624
34.1.3	TCP	626
34.1.4	Blockierende und nicht-blockierende Sockets	628
34.1.5	Erzeugen eines Sockets	629
34.1.6	Die Socket-Klasse	631
34.1.7	Netzwerk-Byte-Order	634
34.1.8	Multiplexende Server – selectors	635
34.1.9	Objektorientierte Serverentwicklung – socketserver	637
34.2	URLs – urllib	639
34.2.1	Zugriff auf entfernte Ressourcen – urllib.request	640
34.2.2	Einlesen und Verarbeiten von URLs – urllib.parse	644
34.3	FTP – ftplib	648
34.3.1	Mit einem FTP-Server verbinden	649
34.3.2	FTP-Kommandos ausführen	650
34.3.3	Mit Dateien und Verzeichnissen arbeiten	650
34.3.4	Übertragen von Dateien	652
34.4	E-Mail	654
34.4.1	SMTP – smtplib	655
34.4.2	POP3 – poplib	658
34.4.3	IMAP4 – imaplib	662
34.4.4	Erstellen komplexer E-Mails – email	668

34.5	Telnet – telnetlib	673
34.5.1	Die Klasse Telnet	673
34.5.2	Beispiel	674
34.6	XML-RPC	676
34.6.1	Der Server	677
34.6.2	Der Client	680
34.6.3	Multicall	682
34.6.4	Einschränkungen	683
35	Debugging und Qualitätssicherung	687
<hr/>		
35.1	Der Debugger	687
35.2	Formatierte Bildschirmausgabe – pprint	690
35.3	Logdateien – logging	691
35.3.1	Das Meldungsformat anpassen	694
35.3.2	Logging Handler	696
35.4	Automatisiertes Testen	698
35.4.1	Testfälle in Docstrings – doctest	698
35.4.2	Unit Tests – unittest	703
35.5	Analyse des Laufzeitverhaltens	706
35.5.1	Laufzeitmessung – timeit	707
35.5.2	Profiling – cProfile	710
35.5.3	Tracing – trace	713
35.6	Optimierung	716
35.6.1	Die Optimize-Option	717
35.6.2	Mutabel vs. immutabel	717
35.6.3	Schleifen	718
35.6.4	Funktionsaufrufe	719
35.6.5	C	719
35.6.6	Lookup	720
35.6.7	Exceptions	720
35.6.8	Keyword Arguments	721
35.6.9	Alternative Interpreter: PyPy	721

36	Dokumentation	723
36.1	Docstrings	723
36.2	Automatisches Erstellen einer Dokumentation – pydoc	725
TEIL V Weiterführende Themen		
37	Anbindung an andere Programmiersprachen	729
37.1	Dynamisch ladbare Bibliotheken – ctypes	730
37.1.1	Ein einfaches Beispiel	730
37.1.2	Die eigene Bibliothek	731
37.1.3	Datentypen	733
37.1.4	Schnittstellenbeschreibung	735
37.1.5	Pointer	737
37.1.6	Strings	738
37.2	Schreiben von Extensions	739
37.2.1	Ein einfaches Beispiel	739
37.2.2	Exceptions	744
37.2.3	Erzeugen der Extension	745
37.2.4	Reference Counting	747
37.3	Python als eingebettete Skriptsprache	748
37.3.1	Ein einfaches Beispiel	748
37.3.2	Ein komplexeres Beispiel	750
37.4	Alternative Interpreter	753
37.4.1	Interoperabilität mit der Java Runtime Environment – Jython	754
37.4.2	Interoperabilität mit .NET – IronPython	759
38	Distribution von Python-Projekten	765
38.1	Eine Geschichte der Distributionen in Python	765
38.1.1	Der klassische Ansatz – distutils	766
38.1.2	Der neue Standard – setuptools	766
38.1.3	Der Paketindex – PyPI und pip	767
38.2	Erstellen von Distributionen – setuptools	767
38.2.1	Schreiben des Moduls	768

38.2.2	Das Installationsskript	769
38.2.3	Erstellen einer Quellcodedistribution	774
38.2.4	Erstellen einer Binärdistribution	774
38.2.5	Distributionen installieren	776
38.2.6	Eigenständige Distributionen erstellen	776
38.2.7	Erstellen von EXE-Dateien – cx_Freeze	777
38.3	Der Python-Paketmanager – pip	778
38.4	Lokalisierung von Programmen – gettext	779
38.4.1	Beispiel für die Verwendung von gettext	780
38.4.2	Erstellen des Sprachkompilats	781
39	Grafische Benutzeroberflächen	785
39.1	Toolkits	785
39.2	Einführung in tkinter	788
39.2.1	Ein einfaches Beispiel	788
39.2.2	Steuerelementvariablen	790
39.2.3	Der Packer	792
39.2.4	Events	796
39.2.5	Steuerelemente	803
39.2.6	Zeichnungen – das Canvas-Widget	823
39.2.7	Weitere Module	830
39.3	Einführung in PyQt	834
39.3.1	Installation	834
39.3.2	Grundlegende Konzepte von Qt	835
39.3.3	Entwicklungsprozess	837
39.4	Signale und Slots	844
39.5	Wichtige Widgets	847
39.5.1	QCheckBox	847
39.5.2	QComboBox	848
39.5.3	QDateEdit, QTimeEdit, QDateTimeEdit	849
39.5.4	QDialog	849
39.5.5	QLineEdit	850
39.5.6	QListWidget, QListView	850
39.5.7	QProgressBar	851
39.5.8	QPushButton	852
39.5.9	QRadioButton	852
39.5.10	QSlider, QDial	852

39.5.11	QTextEdit	853
39.5.12	QWidget	854
39.6	Zeichenfunktionalität	855
39.6.1	Werkzeuge	855
39.6.2	Koordinatensystem	857
39.6.3	Einfache Formen	858
39.6.4	Grafiken	860
39.6.5	Text	861
39.6.6	Eye Candy	863
39.7	Model-View-Architektur	867
39.7.1	Beispielprojekt: ein Adressbuch	868
39.7.2	Auswählen von Einträgen	877
39.7.3	Bearbeiten von Einträgen	879
40	Python als serverseitige Programmiersprache im WWW – ein Einstieg in Django	883
<hr/>		
40.1	Konzepte und Besonderheiten von Django	884
40.2	Installation von Django	885
40.2.1	Installation unter Linux und OS X	886
40.2.2	Installation unter Windows	887
40.3	Erstellen eines neuen Django-Projekts	888
40.3.1	Der Entwicklungswebserver	889
40.3.2	Konfiguration des Projekts	890
40.4	Erstellung einer Applikation	892
40.4.1	Die Applikation in das Projekt einbinden	894
40.4.2	Ein Model definieren	894
40.4.3	Beziehungen zwischen Modellen	895
40.4.4	Übertragung des Modells in die Datenbank	896
40.4.5	Das Model-API	897
40.4.6	Unser Projekt bekommt ein Gesicht	903
40.4.7	Djangos Template-System	910
40.4.8	Verarbeitung von Formulardaten	923
40.4.9	Djangos Administrationsoberfläche	926

41	Wissenschaftliches Rechnen	933
41.1	Installation	934
41.2	Das Modellprogramm	934
41.2.1	Der Import von numpy, scipy und matplotlib	936
41.2.2	Vektorisierung und der Datentyp numpy.ndarray	936
41.2.3	Visualisieren von Daten mit matplotlib.pyplot	940
41.3	Überblick über die Module numpy und scipy	943
41.3.1	Überblick über den Datentyp numpy.ndarray	943
41.3.2	Überblick über scipy	951
42	Insiderwissen	955
42.1	URLs im Standardbrowser öffnen – webbrower	955
42.2	Interpretieren von Binärdaten – struct	955
42.3	Versteckte Passworteingabe	958
42.4	Kommandozeilen-Interpreter	959
42.5	Dateiinterface für Strings – io.StringIO	961
42.6	Generatoren als Konsumenten	962
42.6.1	Ein Decorator für konsumierende Generatorfunktionen	964
42.6.2	Auslösen von Exceptions in einem Generator	965
42.6.3	Eine Pipeline als Verkettung konsumierender Generatorfunktionen	966
42.7	Kopieren von Instanzen – copy	967
42.8	Die interaktive Python-Shell – IPython	971
42.8.1	Die interaktive Shell	971
42.8.2	Das IPython-Notebook	974
42.9	Bildverarbeitung – Pillow	977
42.9.1	Bilddateien laden und speichern	978
42.9.2	Zugriff auf einzelne Pixel	979
42.9.3	Teilbereiche eines Bildes ausschneiden	979
42.9.4	Bilder zusammenfügen	980
42.9.5	Geometrische Bildtransformationen	981
42.9.6	Vordefinierte Bildfilter	982
42.9.7	Eigene Pixeloperationen	983
42.9.8	Bildverbesserungen	984

42.9.9	Zeichenoperationen	985
42.9.10	Interoperabilität	986
43	Von Python 2 nach Python 3	987
<hr/>		
43.1	Die wichtigsten Unterschiede	990
43.1.1	Ein-/Ausgabe	990
43.1.2	Iteratoren	991
43.1.3	Strings	992
43.1.4	Ganze Zahlen	993
43.1.5	Exception Handling	994
43.1.6	Standardbibliothek	994
43.1.7	Neue Sprachelemente in Python 3	995
43.2	Automatische Konvertierung	996
43.3	Geplante Sprachelemente	999
A	Anhang	1001
<hr/>		
A.1	Reservierte Wörter	1001
A.2	Eingebaute Funktionen	1001
A.3	Eingebaute Exceptions	1005
A.4	Python IDEs	1009
Index		1017