

Inhalt

1	Einleitung	27
----------	-------------------	----

2	Die Programmiersprache Python	35
----------	--------------------------------------	----

2.1	Historie, Konzepte, Einsatzgebiete	35
2.1.1	Geschichte und Entstehung	35
2.1.2	Grundlegende Konzepte	36
2.1.3	Einsatzmöglichkeiten und Stärken	38
2.1.4	Einsatzbeispiele	38
2.2	Die Installation von Python	39
2.2.1	Installation von Anaconda unter Windows	40
2.2.2	Installation von Anaconda unter Linux	40
2.2.3	Installation von Anaconda unter macOS	41
2.3	Die Verwendung von Python	41

TEIL I Einstieg in Python

3	Erste Schritte im interaktiven Modus	45
----------	---	----

3.1	Ganze Zahlen	46
3.2	Gleitkommazahlen	47
3.3	Zeichenketten	48
3.4	Listen	48
3.5	Dictionarys	49
3.6	Variablen	50
3.6.1	Die besondere Bedeutung des Unterstrichs	50
3.6.2	Bezeichner	51
3.6.3	Schlüsselwörter und Operatoren	52
3.7	Logische Ausdrücke	52

3.8 Funktionen und Methoden	54
3.8.1 Funktionen	54
3.8.2 Methoden	55
3.9 Bildschirmausgaben	56
3.10 Module	57

4 Der Weg zum ersten Programm 59

4.1 Tippen, kompilieren, testen	59
4.1.1 Shebang	61
4.1.2 Interne Abläufe	61
4.2 Grundstruktur eines Python-Programms	63
4.2.1 Umbrechen langer Zeilen	65
4.2.2 Zusammenfügen mehrerer Zeilen	65
4.3 Das erste Programm	66
4.4 Kommentare	69
4.5 Der Fehlerfall	69

5 Kontrollstrukturen 71

5.1 Fallunterscheidungen	71
5.1.1 Die if-Anweisung	71
5.1.2 Bedingte Ausdrücke	74
5.2 Schleifen	76
5.2.1 Die while-Schleife	76
5.2.2 Abbruch einer Schleife	77
5.2.3 Erkennen eines Schleifenabbruchs	77
5.2.4 Abbruch eines Schleifendurchlaufs	79
5.2.5 Die for-Schleife	81
5.2.6 Die for-Schleife als Zählschleife	82
5.3 Die pass-Anweisung	84

6	Dateien	85
<hr/>		
6.1	Datenströme	85
6.2	Daten aus einer Datei auslesen	86
6.3	Daten in eine Datei schreiben	90
6.4	Das Dateiojekt erzeugen	91
6.4.1	Die Built-in Function open	91
6.4.2	Attribute und Methoden eines Dateiojekts	93
6.4.3	Die Schreib-/Leseposition verändern	94
7	Das Laufzeitmodell	97
<hr/>		
7.1	Die Struktur von Instanzen	99
7.1.1	Datentyp	99
7.1.2	Wert	100
7.1.3	Identität	101
7.2	Referenzen und Instanzen freigeben	103
7.3	Mutable vs. immutable Datentypen	104
7.3.1	Mutable Datentypen und Seiteneffekte	105
8	Funktionen, Methoden und Attribute	109
<hr/>		
8.1	Parameter von Funktionen und Methoden	109
8.1.1	Positionsbezogene Parameter	110
8.1.2	Schlüsselwortparameter	110
8.1.3	Optionale Parameter	111
8.1.4	Reine Schlüsselwortparameter	111
8.2	Attribute	112
9	Informationsquellen zu Python	113
<hr/>		
9.1	Die Built-in Function help	113
9.2	Die Onlinedokumentation	114
9.3	PEPs	114

TEIL II Datentypen

10 Basisdatentypen: eine Übersicht	119
10.1 Das Nichts – NoneType	120
10.2 Operatoren	121
11 Numerische Datentypen	125
11.1 Arithmetische Operatoren	125
11.2 Vergleichende Operatoren	127
11.3 Konvertierung zwischen numerischen Datentypen	128
11.4 Ganzzahlen – int	129
11.4.1 Zahlensysteme	130
11.4.2 Bit-Operationen	131
11.4.3 Methoden	135
11.5 Gleitkommazahlen – float	135
11.6 Boolesche Werte – bool	138
11.6.1 Logische Operatoren	138
11.6.2 Wahrheitswerte nicht-boolescher Datentypen	141
11.6.3 Auswertung logischer Operatoren	142
11.7 Komplexe Zahlen – complex	144
12 Sequenzielle Datentypen	147
12.1 Operationen auf Instanzen sequenzieller Datentypen	148
12.1.1 Ist ein Element vorhanden? – Die Operatoren in und not in	149
12.1.2 Verkettung von Sequenzen – die Operatoren + und +=	151
12.1.3 Wiederholung von Sequenzen – die Operatoren * und *=	152
12.1.4 Zugriff auf bestimmte Elemente einer Sequenz – der []-Operator	153
12.1.5 Länge einer Sequenz – die Built-in Function len	157
12.1.6 Das kleinste und das größte Element einer Sequenz – min und max	158
12.1.7 Die Position eines Elements in der Sequenz – s.index(x, [i, j])	158
12.1.8 Anzahl der Vorkommen eines Elements der Sequenz – s.count(x)	159

12.2 Listen – list	159
12.2.1 Verändern eines Wertes innerhalb der Liste – Zuweisung mit []	161
12.2.2 Ersetzen von Teillisten und Einfügen neuer Elemente – Zuweisung mit []	161
12.2.3 Elemente und Teillisten löschen – del zusammen mit []	162
12.2.4 Methoden von list-Instanzen	162
12.2.5 Weitere Eigenschaften von Listen	168
12.3 Unveränderliche Listen – tuple	171
12.3.1 Packing und Unpacking	172
12.3.2 Immutabel heißt nicht zwingend unveränderlich!	174
12.4 Strings – str, bytes, bytearray	174
12.4.1 Steuerzeichen	177
12.4.2 String-Methoden	179
12.4.3 Formatierung von Strings	190
12.4.4 Zeichensätze und Sonderzeichen	201
13 Zuordnungen und Mengen	211
<hr/>	
13.1 Dictionary – dict	211
13.1.1 Operatoren	214
13.1.2 Methoden	217
13.2 Mengen – set und frozenset	223
13.2.1 Operatoren	225
13.2.2 Methoden	230
13.2.3 Veränderliche Mengen – set	232
13.2.4 Unveränderliche Mengen – frozenset	234
14 Collections	237
<hr/>	
14.1 Verkettete Dictionarys	237
14.2 Zählen von Häufigkeiten	238
14.3 Dictionarys mit Standardwerten	241
14.4 Doppelt verkettete Listen	241
14.5 Benannte Tupel	243

15 Datum und Zeit 247

15.1 Elementare Zeitfunktionen – time 247

 15.1.1 Attribute 249

 15.1.2 Funktionen 250

15.2 Objektorientierte Datumsverwaltung – datetime 255

 15.2.1 datetime.date 256

 15.2.2 datetime.time 257

 15.2.3 datetime.datetime 258

 15.2.4 datetime.timedelta 261

 15.2.5 Operationen für datetime.datetime und datetime.date 264

 15.2.6 Bemerkung zum Umgang mit Zeitzonen 265

16 Weitere Datentypen 267

16.1 Aufzählungstypen – Enum 267

 16.1.1 Aufzählungstyp für Bitmuster – Flag 269

 16.1.2 Ganzzahlige Aufzählungstypen – IntEnum 270

16.2 Datenklassen 271

 16.2.1 Veränderliche Datenklassen 273

 16.2.2 Unveränderliche Datenklassen 274

 16.2.3 Defaultwerte in Datenklassen 275

TEIL III Fortgeschrittene Programmier Techniken

17 Funktionen 279

17.1 Schreiben einer Funktion 281

17.2 Funktionsobjekte 284

17.3 Funktionsparameter 285

 17.3.1 Optionale Parameter 285

 17.3.2 Schlüsselwortparameter 286

 17.3.3 Beliebige Anzahl von Parametern 287

 17.3.4 Reine Schlüsselwortparameter 289

 17.3.5 Reine Positionsparameter 290

17.3.6	Entpacken einer Parameterliste	291
17.3.7	Seiteneffekte	293
17.4	Namensräume	296
17.4.1	Zugriff auf globale Variablen – global	296
17.4.2	Zugriff auf den globalen Namensraum	297
17.4.3	Lokale Funktionen	298
17.4.4	Zugriff auf übergeordnete Namensräume – nonlocal	299
17.4.5	Ungebundene lokale Variablen – eine Stolperfalle	301
17.5	Anonyme Funktionen	302
17.6	Rekursion	303
17.7	Eingebaute Funktionen	304
17.7.1	abs(x)	308
17.7.2	all(iterable)	308
17.7.3	any(iterable)	308
17.7.4	ascii(object)	309
17.7.5	bin(x)	309
17.7.6	bool([x])	309
17.7.7	bytearray([source, encoding, errors])	310
17.7.8	bytes([source, encoding, errors])	311
17.7.9	chr(i)	311
17.7.10	complex([real, imag])	311
17.7.11	dict([source])	312
17.7.12	divmod(a, b)	313
17.7.13	enumerate(iterable[, start])	313
17.7.14	eval(expression, [globals, locals])	313
17.7.15	exec(object, [globals, locals])	314
17.7.16	filter(function, iterable)	314
17.7.17	float([x])	315
17.7.18	format(value, [format_spec])	315
17.7.19	frozenset([iterable])	316
17.7.20	globals()	316
17.7.21	hash(object)	316
17.7.22	help([object])	317
17.7.23	hex(x)	317
17.7.24	id(object)	318
17.7.25	input([prompt])	318
17.7.26	int([x, base])	318
17.7.27	len(s)	319
17.7.28	list([sequence])	319

17.7.29	locals()	320
17.7.30	map(function, [*iterable])	320
17.7.31	max(iterable, {default, key}), max(arg1, arg2, [*args], {key})	321
17.7.32	min(iterable, {default, key}), min(arg1, arg2, [*args], {key})	322
17.7.33	oct(x)	322
17.7.34	ord(c)	322
17.7.35	pow(x, y, [z])	323
17.7.36	print([*objects], {sep, end, file, flush})	323
17.7.37	range([start], stop, [step])	324
17.7.38	repr(object)	325
17.7.39	reversed(sequence)	325
17.7.40	round(x, [n])	325
17.7.41	set([iterable])	326
17.7.42	sorted(iterable, [key, reverse])	326
17.7.43	str([object, encoding, errors])	326
17.7.44	sum(iterable, [start])	327
17.7.45	tuple([iterable])	328
17.7.46	type(object)	328
17.7.47	zip([*iterables])	328

18 Modularisierung 331

18.1	Einbinden globaler Module	331
18.2	Lokale Module	334
18.2.1	Namenskonflikte	335
18.2.2	Modulinterne Referenzen	336
18.2.3	Module ausführen	336
18.3	Pakete	337
18.3.1	Importieren aller Module eines Pakets	339
18.3.2	Namespace Packages	340
18.3.3	Relative Importanweisungen	340
18.4	Das Paket importlib	341
18.4.1	Einbinden von Modulen und Paketen	342
18.4.2	Verändern des Importverhaltens	342

19	Objektorientierung	347
19.1	Klassen	352
19.1.1	Definieren von Methoden	353
19.1.2	Der Konstruktor und die Erzeugung von Attributen	354
19.2	Vererbung	357
19.2.1	Technische Grundlagen	358
19.2.2	Die Klasse GirokontoMitTagesumsatz	361
19.2.3	Mögliche Erweiterungen der Klasse Konto	366
19.2.4	Ausblick	370
19.2.5	Mehrfachvererbung	370
19.3	Setter und Getter und Property-Attribute	372
19.3.1	Setter und Getter	372
19.3.2	Property-Attribute	373
19.4	Klassenattribute und Klassenmethoden sowie statische Methoden	375
19.4.1	Statische Methoden	375
19.4.2	Klassenmethoden	376
19.4.3	Klassenattribute	377
19.5	Built-in Functions für Objektorientierung	378
19.5.1	Funktionen für die Verwaltung der Attribute einer Instanz	378
19.5.2	Funktionen für Informationen über die Klassenhierarchie	380
19.6	Objektphilosophie	381
19.7	Magic Methods und Magic Attributes	383
19.7.1	Allgemeine Magic Methods	383
19.7.2	Operatoren überladen	390
19.7.3	Datentypen emulieren – Duck-Typing	397
20	Ausnahmebehandlung	403
20.1	Exceptions	403
20.1.1	Eingebaute Exceptions	404
20.1.2	Werfen einer Exception	405
20.1.3	Abfangen einer Exception	406
20.1.4	Eigene Exceptions	410
20.1.5	Erneutes Werfen einer Exception	412
20.1.6	Exception Chaining	414

20.2	Zusicherungen – assert	416
20.3	Warnungen	417
21	Iteratoren	419
<hr/>		
21.1	Comprehensions	419
21.1.1	List Comprehensions	419
21.1.2	Dict Comprehensions	421
21.1.3	Set Comprehensions	422
21.2	Generatoren	423
21.2.1	Subgeneratoren	425
21.2.2	Generator Expressions	429
21.3	Iteratoren	430
21.3.1	Verwendung von Iteratoren	433
21.3.2	Mehrere Iteratoren für dieselbe Instanz	436
21.3.3	Nachteile von Iteratoren gegenüber dem direkten Zugriff über Indizes	438
21.3.4	Alternative Definition für iterierbare Objekte	438
21.3.5	Funktionsiteratoren	439
21.4	Spezielle Generatoren – itertools	440
21.4.1	accumulate(iterable, [func])	442
21.4.2	chain([*iterables])	442
21.4.3	combinations(iterable, r)	442
21.4.4	combinations_with_replacement(iterable, r)	443
21.4.5	compress(data, selectors)	443
21.4.6	count([start, step])	444
21.4.7	cycle(iterable)	444
21.4.8	dropwhile(predicate, iterable)	445
21.4.9	filterfalse(predicate, iterable)	445
21.4.10	groupby(iterable, [key])	445
21.4.11	islice(iterable, [start], stop, [step])	446
21.4.12	permutations(iterable, [r])	446
21.4.13	product([*iterables], [repeat])	447
21.4.14	repeat(object, [times])	447
21.4.15	starmap(function, iterable)	448
21.4.16	takewhile(predicate, iterable)	448
21.4.17	tee(iterable, [n])	448
21.4.18	zip_longest([*iterables], {fillvalue})	449

22 Kontextobjekte	451
22.1 Die with-Anweisung	451
22.2 Hilfsfunktionen für with-Kontexte – contextlib	454
22.2.1 Einfache Funktionen als Kontext-Manager	454
22.2.2 Bestimmte Exception-Typen unterdrücken	455
22.2.3 Den Standard-Ausgabestrom umleiten	456
23 Manipulation von Funktionen und Methoden	457
23.1 Decorator	457
23.2 Das Modul functools	460
23.2.1 Funktionsschnittstellen vereinfachen	460
23.2.2 Methodenschnittstellen vereinfachen	462
23.2.3 Caches	463
23.2.4 Ordnungsrelationen vervollständigen	464
23.2.5 Überladen von Funktionen	465
24 Annotationen zur statischen Typprüfung	469
24.1 Annotationen	471
24.1.1 Die Annotation von Funktionen und Methoden	472
24.1.2 Die Annotation von Variablen und Attributen	473
24.1.3 Der Zugriff auf Annotationen zur Laufzeit	475
24.2 Type Hints – das Modul typing	477
24.2.1 Gültige Type Hints	477
24.2.2 Containertypen	478
24.2.3 Typ-Aliasse	479
24.2.4 Type Unions und optionale Werte	480
24.2.5 Typvariablen	481
24.3 Statische Typprüfung in Python – mypy	481
24.3.1 Installation	482
24.3.2 Beispiel	482

TEIL IV Die Standardbibliothek

25	Mathematik	487
<hr/>		
25.1	Mathematische Funktionen – math, cmath	487
25.1.1	Zahlentheoretische Funktionen	488
25.1.2	Exponential- und Logarithmusfunktionen	491
25.1.3	Trigonometrische und hyperbolische Funktionen	491
25.1.4	Distanzen und Normen	492
25.1.5	Umrechnen von Winkeln	492
25.1.6	Darstellungsformen komplexer Zahlen	492
25.2	Zufallszahlengenerator – random	493
25.2.1	Den Status speichern und laden	494
25.2.2	Zufällige ganze Zahlen erzeugen	494
25.2.3	Zufällige Gleitkommazahlen erzeugen	495
25.2.4	Zufallsgesteuerte Operationen auf Sequenzen	495
25.2.5	SystemRandom([seed])	496
25.3	Statistische Berechnungen – statistics	497
25.4	Präzise Dezimalzahlen – decimal	498
25.4.1	Verwendung des Datentyps	499
25.4.2	Nichtnumerische Werte	502
25.4.3	Das Context-Objekt	503
25.5	Hash-Funktionen – hashlib	504
25.5.1	Verwendung des Moduls	506
25.5.2	Weitere Algorithmen	507
25.5.3	Vergleich großer Dateien	507
25.5.4	Passwörter	508
26	Reguläre Ausdrücke	511
<hr/>		
26.1	Syntax regulärer Ausdrücke	511
26.1.1	Beliebige Zeichen	512
26.1.2	Zeichenklassen	512
26.1.3	Quantoren	513
26.1.4	Vordefinierte Zeichenklassen	515
26.1.5	Weitere Sonderzeichen	517
26.1.6	Genügsame Quantoren	518
26.1.7	Gruppen	519

26.1.8	Alternativen	519
26.1.9	Extensions	520
26.2	Verwendung des Moduls	523
26.2.1	Searching	523
26.2.2	Matching	524
26.2.3	Einen String aufspalten	524
26.2.4	Teile eines Strings ersetzen	525
26.2.5	Problematische Zeichen ersetzen	526
26.2.6	Einen regulären Ausdruck kompilieren	526
26.2.7	Flags	526
26.2.8	Das Match-Objekt	528
26.3	Ein einfaches Beispielprogramm – Searching	529
26.4	Ein komplexeres Beispielprogramm – Matching	530
27	Schnittstelle zu Betriebssystem und Laufzeitumgebung	535
<hr/>		
27.1	Funktionen des Betriebssystems – os	535
27.1.1	environ	536
27.1.2	getpid()	536
27.1.3	cpu_count()	536
27.1.4	system(cmd)	537
27.1.5	popen(command, [mode, buffering])	537
27.2	Zugriff auf die Laufzeitumgebung – sys	538
27.2.1	Kommandozeilenparameter	538
27.2.2	Standardpfade	538
27.2.3	Standard-Ein-/Ausgabeströme	539
27.2.4	Das Programm beenden	539
27.2.5	Details zur Python-Version	540
27.2.6	Details zum Betriebssystem	540
27.2.7	Hooks	542
27.3	Kommandozeilenparameter – argparse	544
27.3.1	Taschenrechner – ein einfaches Beispiel	545
27.3.2	Ein weiteres Beispiel	549

28 Dateisystem	553
28.1 Zugriff auf das Dateisystem mit os	553
28.2 Dateipfade – os.path	559
28.3 Zugriff auf das Dateisystem – shutil	564
28.3.1 Verzeichnis- und Dateioperationen	566
28.3.2 Archivoperationen	568
28.4 Temporäre Dateien – tempfile	570
29 Parallele Programmierung	573
29.1 Prozesse, Multitasking und Threads	573
29.1.1 Die Leichtgewichte unter den Prozessen – Threads	574
29.1.2 Threads oder Prozesse?	576
29.1.3 Kooperatives Multitasking – ein dritter Weg	576
29.2 Pythons Schnittstellen zur Parallelisierung	577
29.3 Die abstrakte Schnittstelle – concurrent.futures	578
29.3.1 Ein Beispiel mit einem futures.ThreadPoolExecutor	579
29.3.2 Executor-Instanzen als Kontext-Manager	581
29.3.3 Die Verwendung von futures.ProcessPoolExecutor	582
29.3.4 Die Verwaltung der Aufgaben eines Executors	583
29.4 Die flexible Schnittstelle – threading und multiprocessing	589
29.4.1 Threads in Python – threading	590
29.4.2 Prozesse in Python – multiprocessing	599
29.5 Die kooperative Schnittstelle – asyncio	601
29.5.1 Kooperative Funktionen – Koroutinen	602
29.5.2 Erwartbare Objekte	603
29.5.3 Die Kooperation von Koroutinen – Tasks	604
29.5.4 Ein kooperativer Webcrawler	606
29.5.5 Blockierende Operationen in Koroutinen	615
29.5.6 Weitere asynchrone Sprachmerkmale	617
29.6 Fazit: Welche Schnittstelle ist die richtige?	619

30	Datenspeicherung	621
<hr/>		
30.1	Komprimierte Dateien lesen und schreiben – gzip	621
30.2	XML	623
30.2.1	ElementTree	625
30.2.2	SAX – Simple API for XML	633
30.3	Datenbanken	637
30.3.1	Pythons eingebaute Datenbank – sqlite3	640
30.4	Serialisierung von Instanzen – pickle	657
30.4.1	Funktionale Schnittstelle	658
30.4.2	Objektorientierte Schnittstelle	659
30.5	Das Datenaustauschformat JSON – json	660
30.6	Das Tabellenformat CSV – csv	662
30.6.1	reader-Objekte – Daten aus einer CSV-Datei lesen	663
30.6.2	Dialect-Objekte – eigene Dialekte verwenden	665
31	Netzwerkkommunikation	669
<hr/>		
31.1	Socket API	670
31.1.1	Client-Server-Systeme	671
31.1.2	UDP	674
31.1.3	TCP	675
31.1.4	Blockierende und nichtblockierende Sockets	677
31.1.5	Erzeugen eines Sockets	679
31.1.6	Die Socket-Klasse	680
31.1.7	Netzwerk-Byte-Order	683
31.1.8	Multiplexende Server – selectors	684
31.1.9	Objektorientierte Serverentwicklung – socketserver	687
31.2	URLs – urllib	689
31.2.1	Zugriff auf entfernte Ressourcen – urllib.request	690
31.2.2	Einlesen und Verarbeiten von URLs – urllib.parse	694
31.3	FTP – ftplib	698
31.3.1	Mit einem FTP-Server verbinden	699
31.3.2	FTP-Kommandos ausführen	700
31.3.3	Mit Dateien und Verzeichnissen arbeiten	700
31.3.4	Übertragen von Dateien	701

31.4 E-Mail	704
31.4.1 SMTP – smtp lib	705
31.4.2 POP3 – pop lib	708
31.4.3 IMAP4 – imap lib	712
31.4.4 Erstellen komplexer E-Mails – email	718
31.5 Telnet – telnet lib	722
31.5.1 Die Klasse Telnet	722
31.5.2 Beispiel	723
31.6 XML-RPC	725
31.6.1 Der Server	726
31.6.2 Der Client	730
31.6.3 Multicall	732
31.6.4 Einschränkungen	733

32 Debugging und Qualitätssicherung 737

32.1 Der Debugger	737
32.2 Formatierte Bildschirmausgabe – pprint	740
32.3 Logdateien – logging	742
32.3.1 Das Meldungsformat anpassen	745
32.3.2 Logging-Handler	746
32.4 Automatisiertes Testen	748
32.4.1 Testfälle in Docstrings – doctest	749
32.4.2 Unit Tests – unittest	753
32.5 Analyse des Laufzeitverhaltens	757
32.5.1 Laufzeitmessung – timeit	757
32.5.2 Profiling – cProfile	760
32.5.3 Tracing – trace	764
32.6 Optimierung	766
32.6.1 Die Optimize-Option	767
32.6.2 Mutabel vs. immutabel	768
32.6.3 Funktionsaufrufe	769
32.6.4 Lookups	769

33 Dokumentation 771

- 33.1 Docstrings** 771
- 33.2 Automatisches Erstellen einer Dokumentation – pydoc** 773

TEIL V Weiterführende Themen

34 Alternative Interpreter und Compiler 777

- 34.1 Just-in-Time Kompilierung – PyPy** 777
 - 34.1.1 Installation und Verwendung 778
 - 34.1.2 Beispiel 778
- 34.2 Numba** 778
 - 34.2.1 Installation 779
 - 34.2.2 Beispiel 780
- 34.3 Anbindung an C und C++ – Cython** 781
 - 34.3.1 Installation 781
 - 34.3.2 Die Funktionsweise von Cython 782
 - 34.3.3 Ein Cython-Programm kompilieren 783
 - 34.3.4 Ein Cython-Programm mit statischer Typisierung 785
 - 34.3.5 Eine C-Bibliothek verwenden 786
- 34.4 Die interaktive Python-Shell – IPython** 788
 - 34.4.1 Installation 789
 - 34.4.2 Die interaktive Shell 789
 - 34.4.3 Das Jupyter Notebook 792

35 Distribution von Python-Projekten 797

- 35.1 Eine Geschichte der Distributionen in Python** 797
 - 35.1.1 Der klassische Ansatz – distutils 798
 - 35.1.2 Der neue Standard – setuptools 798
 - 35.1.3 Der Paketindex – PyPI 799
- 35.2 Erstellen von Distributionen – setuptools** 799
 - 35.2.1 Installation 799
 - 35.2.2 Schreiben des Moduls 800
 - 35.2.3 Das Installationskript 801

35.2.4	Erstellen einer Quellcodedistribution	806
35.2.5	Erstellen einer Binärdistribution	806
35.2.6	Distributionen installieren	807
35.3	Erstellen von EXE-Dateien – cx_Freeze	808
35.4	Paketmanager	810
35.4.1	Der Python-Paketmanager – pip	810
35.4.2	Der Paketmanager conda	812
35.5	Virtuelle Umgebungen	814
35.5.1	Das Arbeiten mit virtuellen Umgebungen – venv	815
35.5.2	Virtuelle Umgebungen in Anaconda	817
35.6	Lokalisierung von Programmen – gettext	818
35.6.1	Beispiel für die Verwendung von gettext	819
35.6.2	Erstellen des Sprachkompilats	820
36	Grafische Benutzeroberflächen	823
<hr/>		
36.1	Toolkits	823
36.2	Einführung in tkinter	825
36.2.1	Ein einfaches Beispiel	826
36.2.2	Steuerelementvariablen	828
36.2.3	Der Packer	830
36.2.4	Events	834
36.2.5	Steuerelemente	841
36.2.6	Zeichnungen – das Canvas-Widget	859
36.2.7	Weitere Module	868
36.3	Einführung in PySide2	871
36.3.1	Installation	872
36.3.2	Grundlegende Konzepte von Qt	872
36.3.3	Entwicklungsprozess	874
36.4	Signale und Slots	881
36.5	Wichtige Widgets	884
36.5.1	QCheckBox	884
36.5.2	QComboBox	885
36.5.3	QDateEdit, QTimeEdit, QDateTimeEdit	886
36.5.4	QDialog	886
36.5.5	QLineEdit	887
36.5.6	QListWidget, QListView	887

36.5.7	QProgressBar	888
36.5.8	QPushButton	889
36.5.9	QRadioButton	889
36.5.10	QSlider, QDial	889
36.5.11	QTextEdit	890
36.5.12	QWidget	891
36.6	Zeichenfunktionalität	892
36.6.1	Werkzeuge	893
36.6.2	Koordinatensystem	895
36.6.3	Einfache Formen	895
36.6.4	Grafiken	898
36.6.5	Text	899
36.6.6	Eye Candy	900
36.7	Model-View-Architektur	904
36.7.1	Beispielprojekt: ein Adressbuch	905
36.7.2	Auswählen von Einträgen	914
36.7.3	Bearbeiten von Einträgen	916

37 Python als serverseitige Programmiersprache im WWW – ein Einstieg in Django 921

37.1	Konzepte und Besonderheiten von Django	922
37.2	Installation von Django	923
37.3	Erstellen eines neuen Django-Projekts	924
37.3.1	Der Entwicklungswebserver	925
37.3.2	Konfiguration des Projekts	927
37.4	Erstellung einer Applikation	928
37.4.1	Die Applikation in das Projekt einbinden	930
37.4.2	Ein Model definieren	930
37.4.3	Beziehungen zwischen Modellen	931
37.4.4	Übertragung des Modells in die Datenbank	932
37.4.5	Die Model-API	933
37.4.6	Unser Projekt bekommt ein Gesicht	939
37.4.7	Djangos Template-System	946
37.4.8	Verarbeitung von Formulardaten	959
37.4.9	Djangos Administrationsoberfläche	962

38	Wissenschaftliches Rechnen und Data Science	969
<hr/>		
38.1	Installation	970
38.2	Das Modellprogramm	971
38.2.1	Der Import von numpy, scipy und matplotlib	972
38.2.2	Vektorisierung und der Datentyp numpy.ndarray	973
38.2.3	Visualisieren von Daten mit matplotlib.pyplot	976
38.3	Überblick über die Module numpy und scipy	979
38.3.1	Überblick über den Datentyp numpy.ndarray	980
38.3.2	Überblick über scipy	988
38.4	Eine Einführung in die Datenanalyse mit pandas	990
38.4.1	Das DataFrame-Objekt	991
38.4.2	Selektiver Datenzugriff	992
38.4.3	Löschen von Zeilen und Spalten	998
38.4.4	Einfügen von Zeilen und Spalten	999
38.4.5	Logische Ausdrücke auf Datensätzen	999
38.4.6	Manipulation von Datensätzen	1001
38.4.7	Ein- und Ausgabe	1003
38.4.8	Visualisierung	1003
39	Insiderwissen	1007
<hr/>		
39.1	Zuweisungsausdrücke	1007
39.1.1	Motivation	1008
39.2	URLs im Standardbrowser öffnen – webbrower	1009
39.3	Interpretieren von Binärdaten – struct	1010
39.4	Versteckte Passworteingabe	1012
39.5	Kommandozeilen-Interpreter	1013
39.6	Dateiinterface für Strings – io.StringIO	1016
39.7	Generatoren als Konsumenten	1016
39.7.1	Ein Decorator für konsumierende Generatorfunktionen	1018
39.7.2	Auslösen von Exceptions in einem Generator	1019
39.7.3	Eine Pipeline als Verkettung konsumierender Generatorfunktionen	1020
39.8	Kopieren von Instanzen – copy	1022
39.9	Bildverarbeitung – Pillow	1025
39.9.1	Installation	1025

39.9.2	Bilddateien laden und speichern	1026
39.9.3	Zugriff auf einzelne Pixel	1027
39.9.4	Manipulation von Bildern	1027
39.9.5	Interoperabilität	1034
40	Von Python 2 nach Python 3	1035
40.1	Die wichtigsten Unterschiede	1038
40.1.1	Ein-/Ausgabe	1039
40.1.2	Iteratoren	1040
40.1.3	Strings	1040
40.1.4	Ganze Zahlen	1042
40.1.5	Exception Handling	1042
40.1.6	Standardbibliothek	1043
40.2	Automatische Konvertierung	1044
40.3	Geplante Sprachelemente	1047
A	Anhang	1049
A.1	Reservierte Wörter	1049
A.2	Operatorrangfolge	1049
A.3	Eingebaute Funktionen	1051
A.4	Eingebaute Exceptions	1055
A.5	Python IDEs	1059
Index	1063