

# Inhaltsverzeichnis

<b>1</b>	<b>Python installieren</b>	<b>1</b>
1.1	Los geht's	1
1.2	Welche Python-Version ist die Richtige für Sie?	1
1.3	Installation unter Microsoft Windows	2
1.4	Installation unter Mac OS X	4
1.5	Installation unter Ubuntu Linux	6
1.6	Installation auf anderen Plattformen	7
1.7	Verwenden der Python-Shell	7
1.8	Python-Editoren und -IDEs	10
<b>2</b>	<b>Ihr erstes Python-Programm</b>	<b>11</b>
2.1	Los geht's	11
2.2	Funktionen deklarieren	12
2.2.1	Pythons Datentypen im Vergleich mit denen anderer Sprachen	13
2.3	Lesbaren Code schreiben	14
2.3.1	Docstrings	14
2.4	Der import-Suchpfad	15
2.5	Alles ist ein Objekt	16
2.5.1	Was ist ein Objekt?	16
2.6	Code einrücken	17
2.7	Ausnahmen	18
2.7.1	Importfehler abfangen	19
2.8	Ungebundene Variablen	20
2.9	Groß- und Kleinschreibung bei Namen	21
2.10	Skripte ausführen	21
<b>3</b>	<b>Native Datentypen</b>	<b>23</b>
3.1	Los geht's	23
3.2	Boolesche Werte	23
3.3	Zahlen	24
3.3.1	int- in float-Werte umwandeln und anders herum	25
3.3.2	Einfache Rechenoperationen	26

3.3.3	Brüche	27
3.3.4	Trigonometrie	27
3.3.5	Zahlen in einem booleschen Kontext	28
3.4	Listen	29
3.4.1	Erstellen einer Liste	29
3.4.2	Slicing einer Liste	30
3.4.3	Elemente zu einer Liste hinzufügen	31
3.4.4	Innerhalb einer Liste nach Werten suchen	33
3.4.5	Elemente aus einer Liste entfernen	34
3.4.6	Elemente aus einer Liste entfernen: Bonusrunde	34
3.4.7	Listen in einem booleschen Kontext	35
3.5	Tupel	36
3.5.1	Tupel in einem booleschen Kontext	38
3.5.2	Mehrere Werte auf einmal zuweisen	38
3.6	Sets	39
3.6.1	Ein Set erstellen	39
3.6.2	Ein Set verändern	41
3.6.3	Elemente aus einem Set entfernen	42
3.6.4	Einfache Mengenoperationen	43
3.6.5	Sets in einem booleschen Kontext	45
3.7	Dictionaries	46
3.7.1	Erstellen eines Dictionarys	46
3.7.2	Ein Dictionary verändern	47
3.7.3	Dictionarys mit gemischten Werten	48
3.7.4	Dictionarys in einem booleschen Kontext	49
3.8	None	49
3.8.1	None in einem booleschen Kontext	50
4	<b>Comprehensions</b>	51
4.1	Los geht's	51
4.2	Mit Dateien und Verzeichnissen arbeiten	51
4.2.1	Das aktuelle Arbeitsverzeichnis	51
4.2.2	Mit Dateinamen und Verzeichnisnamen arbeiten	52
4.2.3	Verzeichnisse auflisten	54
4.2.4	Metadaten von Dateien erhalten	55
4.2.5	Absolute Pfadnamen erstellen	56
4.3	List Comprehensions	56
4.4	Dictionary Comprehensions	58
4.4.1	Andere tolle Sachen, die man mit Dictionary Comprehensions machen kann	60
4.5	Set Comprehensions	60
5	<b>Strings</b>	61
5.1	Langweiliges Zeug, das Sie wissen müssen, bevor es losgeht	61
5.2	Unicode	63
5.3	Los geht's	65

5.4	Strings formatieren	66
5.4.1	Zusammengesetzte Feldnamen	67
5.4.2	Formatmodifizierer	69
5.5	Andere häufig verwendete String-Methoden	69
5.5.1	Slicen eines Strings	71
5.6	Strings vs. Bytes	72
5.7	Nachbemerkung – Zeichencodierung von Python-Quelltext	75
<b>6</b>	<b>Reguläre Ausdrücke</b>	<b>77</b>
6.1	Los geht's	77
6.2	Fallbeispiel: Adresse	78
6.3	Fallbeispiel: römische Zahlen	80
6.3.1	Prüfen der Tausender	81
6.3.2	Prüfen der Hunderter	82
6.4	Verwenden der {n,m}-Syntax	84
6.4.1	Prüfen der Zehner und Einer	85
6.5	Ausführliche reguläre Ausdrücke	87
6.6	Fallbeispiel: Telefonnummern gliedern	88
6.7	Zusammenfassung	94
<b>7</b>	<b>Closures und Generatoren</b>	<b>95</b>
7.1	Abtauchen	95
7.2	Nutzen wir reguläre Ausdrücke!	96
7.3	Eine Funktionsliste	98
7.4	Eine Musterliste	101
7.5	Eine Musterdatei	103
7.6	Generatoren	104
7.6.1	Ein Fibonacci-Generator	106
7.6.2	Ein Generator für Plural-Regeln	107
<b>8</b>	<b>Klassen und Iteratoren</b>	<b>109</b>
8.1	Los geht's	109
8.2	Klassen definieren	110
8.2.1	Die <code>__init__()</code> -Methode	110
8.3	Klassen instanziiieren	111
8.4	Instanzvariablen	112
8.5	Ein Fibonacci-Iterator	113
8.6	Ein Iterator für Plural-Regeln	115
<b>9</b>	<b>Erweiterte Iteratoren</b>	<b>121</b>
9.1	Los geht's	121
9.2	Alle Vorkommen eines Musters finden	123
9.3	Die einmaligen Elemente einer Folge finden	124
9.4	Bedingungen aufstellen	124
9.5	Generator-Ausdrücke	125
9.6	Permutationen berechnen ... Auf die faule Art!	126

9.7	Anderes cooles Zeug im Modul <code>itertools</code> .....	128
9.8	Eine neue Art der String-Manipulation .....	132
9.9	Herausfinden, ob ein beliebiger String ein Python-Ausdruck ist .....	134
9.10	Alles zusammenfügen .....	137
<b>10</b>	<b>Unit Testing</b> .....	139
10.1	Los geht's (noch nicht) .....	139
10.2	Eine Frage .....	140
10.3	Anhalten und Alarm schlagen .....	146
10.4	Wieder anhalten und wieder Alarm .....	150
10.5	Noch eine Kleinigkeit .....	152
10.6	Eine erfreuliche Symmetrie .....	155
10.7	Noch mehr schlechte Eingaben .....	158
<b>11</b>	<b>Refactoring</b> .....	163
11.1	Los geht's .....	163
11.2	Mit sich ändernden Anforderungen umgehen .....	166
11.3	Refactoring .....	170
11.4	Zusammenfassung .....	174
<b>12</b>	<b>Dateien</b> .....	177
12.1	Los geht's .....	177
12.2	Aus Textdateien lesen .....	177
12.2.1	Die Zeichencodierung zeigt ihre hässliche Fratze .....	178
12.2.2	Streamobjekte .....	179
12.2.3	Daten aus einer Textdatei lesen .....	180
12.2.4	Dateien schließen .....	182
12.2.5	Automatisches Schließen von Dateien .....	183
12.2.6	Daten zeilenweise lesen .....	184
12.3	In Textdateien schreiben .....	185
12.3.1	Schon wieder Zeichencodierung .....	186
12.4	Binärdateien .....	187
12.5	Streamobjekte aus anderen Quellen als Dateien .....	188
12.5.1	Umgang mit komprimierten Dateien .....	189
12.6	Standardeingabe, -ausgabe und -fehler .....	191
12.6.1	Die Standardausgabe umleiten .....	192
<b>13</b>	<b>XML</b> .....	195
13.1	Los geht's .....	195
13.2	Ein XML-Crashkurs .....	197
13.3	Der Aufbau eines Atom-Feeds .....	199
13.4	XML parsen .....	201
13.4.1	Elemente sind Listen .....	202
13.4.2	Attribute sind Dictionarys .....	203
13.5	Innerhalb eines XML-Dokuments nach Knoten suchen .....	204

13.6	Noch mehr XML .....	207
13.7	XML erzeugen .....	209
13.8	Beschädigtes XML parsen .....	212
<b>14</b>	<b>Python-Objekte serialisieren .....</b>	<b>215</b>
14.1	Los geht's .....	215
14.1.1	Eine kurze Bemerkung zu den Beispielen dieses Kapitels .....	216
14.2	Daten in einer pickle-Datei speichern .....	216
14.3	Daten aus einer pickle-Datei lesen .....	218
14.4	pickle ohne Datei .....	219
14.5	Bytes und Strings zeigen ein weiteres Mal ihre hässlichen Fratzen .....	220
14.6	pickle-Dateien debuggen .....	220
14.7	Serialisierte Python-Objekte in anderen Sprachen lesbar machen .....	223
14.8	Daten in einer JSON-Datei speichern .....	223
14.9	Entsprechungen der Python-Datentypen in JSON .....	225
14.10	Von JSON nicht unterstützte Datentypen serialisieren .....	226
14.11	Daten aus einer JSON-Datei laden .....	229
<b>15</b>	<b>HTTP-Webdienste .....</b>	<b>233</b>
15.1	Los geht's .....	233
15.2	Eigenschaften von HTTP .....	234
15.2.1	Caching .....	234
15.2.2	Überprüfen des Datums der letzten Änderung .....	236
15.2.3	ETags .....	237
15.2.4	Komprimierung .....	238
15.2.5	Weiterleitungen .....	238
15.3	Wie man Daten nicht über HTTP abrufen sollte .....	239
15.4	Was geht über's Netz .....	240
15.5	Vorstellung von httpplib2 .....	243
15.5.1	Ein kleiner Exkurs zur Erklärung, warum httpplib2 Bytes statt Strings zurückgibt .....	246
15.5.2	Wie httpplib2 mit Caching umgeht .....	247
15.5.3	Wie httpplib2 mit Last-Modified- und ETag- Headern umgeht .....	250
15.5.4	Wie httpplib2 mit Komprimierung umgeht .....	252
15.5.5	Wie httpplib2 mit Weiterleitungen umgeht .....	253
15.6	Über HTTP-GET hinaus .....	257
15.7	Über HTTP-POST hinaus .....	260
<b>16</b>	<b>Fallstudie: chardet zu Python 3 portieren .....</b>	<b>263</b>
16.1	Los geht's .....	263
16.2	Was ist die automatische Zeichencodierungserkennung? .....	263
16.2.1	Ist das nicht unmöglich? .....	263
16.2.2	Existiert solch ein Algorithmus? .....	264

16.3	Das chardet-Modul .....	264
16.3.1	UTF-n mit einer Byte Order Mark .....	265
16.3.2	Escape-Codierungen .....	265
16.3.3	Multi-Byte-Codierungen .....	265
16.3.4	Single-Byte-Codierungen .....	266
16.3.5	windows-1252 .....	267
16.4	2to3 ausführen .....	267
16.5	Mehr-Dateien-Module .....	270
16.6	Anpassen, was 2to3 nicht anpassen kann .....	272
16.6.1	False ist ungültige Syntax .....	272
16.6.2	Kein Modul namens constants .....	273
16.6.3	Bezeichner 'file' ist nicht definiert .....	274
16.6.4	Ein Stringmuster kann nicht auf ein bytearrayiges Objekt angewandt werden .....	274
16.6.5	Implizite Umwandlung eines 'bytes'-Objekts in str nicht möglich .....	276
16.6.6	Nicht unterstützte Datentypen für Operand +: 'int' und 'bytes' .....	278
16.6.7	ord() erwartet String der Länge 1, int gefunden ....	280
16.6.8	Unsortierbare Datentypen: int() >= str() .....	282
16.6.9	Globaler Bezeichner 'reduce' ist nicht definiert ....	284
16.7	Zusammenfassung .....	286
<b>17</b>	<b>Python-Bibliotheken packen .....</b>	<b>287</b>
17.1	Los geht's .....	287
17.2	Was kann Distutils nicht für Sie tun? .....	288
17.3	Verzeichnisstruktur .....	289
17.4	Das Setup-Skript schreiben .....	291
17.5	Ihr Paket klassifizieren .....	292
17.5.1	Beispiele guter Paket-Klassifizierer .....	293
17.6	Zusätzliche Dateien mit einem Manifest angeben .....	294
17.7	Ihr Setup-Skript auf Fehler untersuchen .....	295
17.8	Eine Quellcode-Distribution erstellen .....	296
17.9	Einen grafischen Installer erstellen .....	298
17.9.1	Installierbare Pakete für andere Betriebssysteme erzeugen. ....	299
17.10	Ihre Software zum Python Package Index hinzufügen .....	299
17.11	Die Zukunft des Packens von Python-Software .....	301
<b>Anhang A – Code mithilfe von 2to3 von Python 2 zu</b>		
<b>Python 3</b>	<b>portieren .....</b>	<b>303</b>
A.1	Los geht's .....	303
A.2	print-Anweisung .....	303
A.3	Unicode-Stringlitterale .....	304
A.4	Globale unicode()-Funktion .....	304

A.5	Datentyp <code>long</code> .....	304
A.6	<code>&lt;&gt;</code> -Vergleich .....	305
A.7	Dictionary-Methode <code>has_key()</code> .....	305
A.8	Dictionary-Methoden, die Listen zurückgeben .....	306
A.9	Umbenannte und umstrukturierte Module .....	307
A.9.1	<code>http</code> .....	307
A.9.2	<code>urllib</code> .....	308
A.9.3	<code>dbm</code> .....	309
A.9.4	<code>xmlrpc</code> .....	309
A.9.5	Weitere Module .....	309
A.10	Relative Importe innerhalb eines Pakets .....	310
A.11	Die Iteratormethode <code>next()</code> .....	312
A.12	Die globale Funktion <code>filter()</code> .....	312
A.13	Die globale Funktion <code>map()</code> .....	313
A.14	Die globale Funktion <code>reduce()</code> .....	314
A.15	Die globale Funktion <code>apply()</code> .....	314
A.16	Die globale Funktion <code>intern()</code> .....	315
A.17	<code>exec</code> -Anweisung .....	315
A.18	<code>execfile</code> -Anweisung .....	316
A.19	<code>repr</code> -Literale (Backticks) .....	316
A.20	<code>try...except</code> -Anweisung .....	316
A.21	<code>raise</code> -Anweisung .....	317
A.22	<code>throw</code> -Methode bei Generatoren .....	318
A.23	Die globale Funktion <code>xrange()</code> .....	318
A.24	Die globalen Funktionen <code>raw_input()</code> und <code>input()</code> .....	319
A.25	<code>func *</code> -Funktionsattribute .....	320
A.26	Die Ein-/Ausgabemethode <code>xreadlines()</code> .....	320
A.27	<code>lambda</code> -Funktionen, die ein Tupel anstatt mehrerer Parameter übernehmen .....	321
A.28	Besondere Methodenattribute .....	322
A.29	Die spezielle Methode <code>__nonzero__</code> .....	322
A.30	Oktale Literale .....	323
A.31	<code>sys.maxint</code> .....	323
A.32	Die globale Funktion <code>callable()</code> .....	323
A.33	Die globale Funktion <code>zip()</code> .....	323
A.34	Die Ausnahme <code>StandardError</code> .....	324
A.35	Konstanten des Moduls <code>types</code> .....	324
A.36	Die globale Funktion <code>isinstance()</code> .....	325
A.37	Der Datentyp <code>basestring</code> .....	325
A.38	Das Modul <code>itertools</code> .....	326
A.39	<code>sys.exc_type</code> , <code>sys.exc_value</code> , <code>sys.exc_traceback</code> .....	326
A.40	Tupeldurchlaufende List Comprehensions .....	327
A.41	Die Funktion <code>os.getcwd()</code> .....	327

A.42	Metaklassen .....	327
A.43	Stilfragen .....	328
A.43.1	<code>set()</code> -Literale (ausdrücklich) .....	328
A.43.2	Die globale Funktion <code>buffer()</code> (ausdrücklich) .....	328
A.43.3	Whitespace bei Kommas (ausdrücklich) .....	329
A.43.4	Geläufige Ausdrücke (ausdrücklich) .....	329
<b>Anhang B</b>	<b>– Spezielle Methoden .....</b>	<b>331</b>
B.1	Los geht's .....	331
B.2	Grundlegendes .....	331
B.3	Klassen, die sich wie Iteratoren verhalten .....	332
B.4	Berechnete Attribute .....	333
B.5	Klassen, die sich wie Funktionen verhalten .....	335
B.6	Klassen, die sich wie Folgen verhalten .....	337
B.7	Klassen, die sich wie Dictionaries verhalten .....	338
B.8	Klassen, die sich wie Zahlen verhalten .....	339
B.9	Vergleichbare Klassen .....	342
B.10	Serialisierbare Klassen .....	343
B.11	Klassen, die innerhalb eines <code>with</code> -Blocks verwendet werden können .....	343
B.12	Wirklich seltsames Zeug .....	344
<b>Sachverzeichnis</b> .....		<b>347</b>