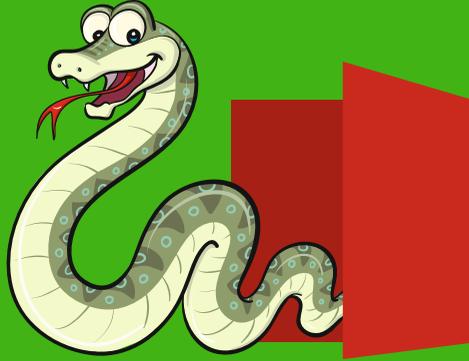


Bernd KLEIN

MIT CODE-
HIGHLIGHTING



EINFÜHRUNG IN PYTHON 3

FÜR EIN- UND UMSTEIGER

4. Auflage



Im Internet:
Musterlösungen zu den Übungen

HANSER

Klein
Einführung in Python 3



Bleiben Sie auf dem Laufenden!

Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter:

www.hanser-fachbuch.de/newsletter



Bernd Klein

Einführung in Python 3

Für Ein- und Umsteiger

4., vollständig überarbeitete Auflage

HANSER

Der Autor:

Bernd Klein, bernd@python-kurs.eu

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autor und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2021 Carl Hanser Verlag München, www.hanser-fachbuch.de

Lektorat: Brigitte Bauer-Schiewek

Copy editing: Jürgen Dubau, Freiburg/Elbe

Layout: le-tex publishing services, Leipzig

Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München

Umschlagrealisation: Max Kostopoulos

Titelmotiv: © istockphoto.com/zaricm

Druck und Bindung: Eberl & Koesel GmbH & Co. KG, Krugzell

Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702

Printed in Germany

Print-ISBN: 978-3-446-46379-0

E-Book-ISBN: 978-3-446-46556-5

E-Pub-ISBN: 978-3-446-46467-4

Inhalt

Vorwort	XIX
Danksagung	XX
Teil I: Einleitung	1
1 Einleitung	3
1.1 Einfach und schnell zu lernen	3
1.2 Geschichte von Python	3
1.3 Zen von Python	4
1.4 Zielgruppe des Buches	5
1.5 Aufbau des Buches	6
1.6 Programmieren lernen „interaktiv“	7
1.7 Download der Beispiele und Hilfe	8
1.8 Anregungen und Kritik	8
Teil II: Grundlagen	9
2 Kommandos und Programme	11
2.1 Erste Schritte mit Python	11
2.1.1 Linux	11
2.1.2 Windows	12
2.1.3 macOS	13
2.2 Herkunft und Bedeutung des Begriffes interaktive Shell	13
2.2.1 Erste Schritte in der interaktiven Shell	14
2.3 Verlassen der Python-Shell	15
2.4 Benutzung von Variablen	15
2.5 Mehrzeilige Anweisungen in der interaktiven Shell	16
2.6 Programme schreiben oder schnell mal der Welt “Hallo” sagen	17

3	Bytecode und Maschinencode	21
3.1	Einführung.....	21
3.2	Unterschied zwischen Programmier- und Skriptsprachen	21
3.3	Interpreter- oder Compilersprache.....	21
4	Datentypen und Variablen	25
4.1	Einführung.....	25
4.2	Variablennamen	28
	4.2.1 Gültige Variablennamen	28
	4.2.2 Konventionen für Variablennamen	29
4.3	Datentypen	29
	4.3.1 Ganze Zahlen	29
	4.3.2 Fließkommazahlen.....	31
	4.3.3 Zeichenketten.....	31
	4.3.4 Boolesche Werte	31
	4.3.5 Komplexe Zahlen	32
	4.3.6 Operatoren	32
4.4	Statische und dynamische Typdeklaration	34
4.5	Typumwandlung	35
4.6	Datentyp ermitteln	36
5	Sequentielle Datentypen	39
5.1	Übersicht.....	39
	5.1.1 Zeichenketten oder Strings	40
	5.1.2 Listen.....	42
	5.1.3 Tupel	42
5.2	Indizierung von sequentiellen Datentypen	43
5.3	Teilbereichsoperator.....	44
5.4	Die len-Funktion	47
5.5	Aufgaben	47
6	Listen und Tupel im Detail.....	49
6.1	Virtueller Einkaufsbummel.....	49
6.2	Stapelspeicher/Stacks	51
6.3	Stapelverarbeitung in Python: pop und append.....	52
6.4	extend	52
6.5	Module importieren	53
6.6	,+'-Operator oder append	55
6.7	Entfernen eines Wertes.....	56

6.8	Prüfen, ob ein Element in Liste enthalten ist	57
6.9	Finden der Position eines Elementes	57
6.10	Einfügen von Elementen	57
6.11	Besonderheiten bei Tupel	58
6.11.1	Leere Tupel	58
6.11.2	1-Tupel	58
6.11.3	Mehrfachzuweisungen, Packing und Unpacking	59
6.12	Die veränderliche Unveränderliche	60
6.13	Aufgaben	60
7	Verzweigungen	63
7.1	Anweisungsblöcke und Einrückungen	63
7.2	Bedingte Anweisungen in Python	66
7.2.1	Einfachste if-Anweisung	66
7.2.2	if-Anweisung mit else-Zweig	67
7.2.3	elif-Zweige	67
7.3	Vergleichsoperatoren	68
7.4	Zusammengesetzte Bedingungen	68
7.5	Wahr oder falsch: Bedingungen in Verzweigungen	69
7.6	Aufgaben	70
8	Schleifen	71
8.1	Übersicht	71
8.2	while-Schleife	72
8.3	break und continue	73
8.4	Die Alternative im Erfolgsfall: else	74
8.5	Iterierbare Objekte (iterables)	76
8.6	For-Schleife	77
8.7	Aufgaben	80
9	Dictionaries	83
9.1	Definition und Benutzung	83
9.2	Fehlerfreie Zugriffe auf Dictionaries	86
9.3	Einfachere Definition von Dictionaries	88
9.4	Zulässige Typen für Schlüssel und Werte	88
9.5	Verschachtelte Dictionaries	89
9.6	Dictionaries in Listen wandeln	89
9.7	Weitere Methoden auf Dictionaries	90
9.8	Operatoren	92

9.9	Die zip-Funktion	93
9.10	Dictionaries aus Listen erzeugen	95
9.11	Aufgaben	95
10	Mengen	99
10.1	Übersicht	99
10.2	Mengen in Python	99
10.2.1	Sets erzeugen	100
10.2.2	Mengen von unveränderlichen Elementen	100
10.3	Frozensets	101
10.4	Operationen auf „set“-Objekten	101
10.4.1	add(element)	101
10.4.2	clear()	101
10.4.3	copy	102
10.4.4	difference()	102
10.4.5	difference_update()	102
10.4.6	discard(el)	103
10.4.7	remove(el)	103
10.4.8	intersection(s)	103
10.4.9	union(s)	104
10.4.10	isdisjoint()	104
10.4.11	issubset()	104
10.4.12	issuperset()	105
10.4.13	pop()	105
10.5	Erweiterte Zuweisungsoperatoren für Mengen	106
11	Eingaben	107
11.1	Eingabe mittels input	107
12	Dateien lesen und schreiben	109
12.1	Dateien	109
12.2	Text aus einer Datei lesen	109
12.3	Schreiben in eine Datei	111
12.4	In einem Rutsch lesen: readlines und read	111
12.5	with-Anweisung	112
12.6	Aufgaben	113

13	Formatierte Ausgabe und Strings formatieren	115
13.1	Wege, die Ausgabe zu formatieren	115
13.2	print-Funktion	116
13.3	Notwendigkeit	118
13.4	Formatierte Stringlitterale / f-Strings	118
13.5	Die String-Methode „format“	123
13.6	Stringmodulo-Operator oder Formatierung à la C	125
13.7	Benutzung von Dictionaries beim Aufruf der „format“-Methode	129
13.8	Benutzung von lokalen Variablen in „format“	130
13.9	Weitere String-Methoden zum Formatieren	131
14	Referenzen, flaches und tiefes Kopieren	133
14.1	Einführung	133
14.2	Swen und Sarah	133
14.3	Variablen sind Referenzen	134
14.4	Kopieren einer Liste	135
14.5	Flache Kopien	136
14.6	Kopieren mit deepcopy	138
14.7	Problemverhinderung	139
14.8	Deepcopy für Dictionaries	139
15	Funktionen	141
15.1	Allgemein	141
15.2	Funktionen	141
15.3	Docstring	143
15.4	Standardwerte für Funktionen	145
15.5	Schlüsselwortparameter	146
15.6	Funktionen ohne oder mit leerer return-Anweisung	146
15.7	Mehrere Rückgabewerte	147
15.8	Parameterübergabe im Detail	148
15.9	Effekte bei veränderlichen Objekten	150
15.10	Kommandozeilenparameter	151
15.11	Variable Anzahl von Parametern / Variadische Funktionen	152
15.12	* in Funktionsaufrufen	154
15.13	Beliebige Schlüsselwortparameter	155
15.14	Doppeltes Sternchen im Funktionsaufruf	155
15.15	Aufgaben	155

16	Wertebereich von Variablen	159
16.1	Einführung	159
16.2	Globale und lokale Variablen in Funktionen	159
16.3	nonlocal-Variablen	162
17	Rekursive Funktionen	167
17.1	Definition und Herkunft des Begriffs	167
17.2	Definition der Rekursion	168
17.3	Rekursive Funktionen in Python	168
17.4	Die Tücken der Rekursion	169
17.5	Fibonacci-Folge in Python	170
17.6	Aufgaben	174
18	Sortieren	177
18.1	Sortieren von Listen	177
18.1.1	„sort“ und „sorted“	177
18.1.2	Umkehrung der Sortierreihenfolge	178
18.1.3	Eigene Sortierfunktionen	178
18.2	Aufgaben	181
19	Modularisierung	183
19.1	Module	183
19.1.1	Namensräume von Modulen	184
19.1.2	Namensräume umbenennen	185
19.1.3	Modularten	185
19.1.4	Suchpfad für Module	186
19.1.5	Inhalt eines Moduls	187
19.1.6	Eigene Module	187
19.1.7	Dokumentation für eigene Module	188
19.2	Pakete	189
19.2.1	Einfaches Paket erzeugen	190
19.2.2	Komplexeres Paket	191
19.2.3	Komplettes Paket importieren	193
20	Alles über Strings	197
20.1	... fast alles	197
20.2	Aufspalten von Zeichenketten	198
20.2.1	split	199
20.2.2	rsplit	201

20.2.3	Folge von Trennzeichen	202
20.2.4	splitlines	203
20.2.5	partition und rpartition	204
20.3	Zusammenfügen von Stringlisten mit join	204
20.4	Suchen von Teilstrings	205
20.4.1	„in“ oder „not in“	205
20.4.2	s.find(substring[, start[, end]])	205
20.4.3	s.rfind(substring[, start[, end]])	206
20.4.4	s.index(substring[, start[, end]])	206
20.4.5	s.rindex(substring[, start[, end]])	206
20.4.6	s.count(substring[, start[, end]])	206
20.5	Suchen und Ersetzen	207
20.6	Nur noch Kleinbuchstaben oder Großbuchstaben	207
20.7	capitalize und title	208
20.8	Stripping Strings	208
20.9	Strings ausrichten	209
20.10	String-Tests	209
20.11	Aufgaben	211
21	Ausnahmebehandlung	215
21.1	Abfangen mehrerer Ausnahmen	217
21.2	except mit mehrfachen Ausnahmen	218
21.3	Die optionale else-Klausel	218
21.4	Fehlerinformationen über sys.exc_info	219
21.5	Exceptions generieren	220
21.6	Finalisierungsaktion	220
Teil III:	Objektorientierte Programmierung	223
22	Grundlegende Aspekte	225
22.1	Bibliotheksvergleich	225
22.2	Objekte und Instanzen einer Klasse	227
22.3	Kapselung von Daten und Methoden	228
22.4	Eine minimale Klasse in Python	228
22.5	Eigenschaften und Attribute	229
22.6	Methoden	232
22.7	Instanzvariablen	232
22.8	Die __init__-Methode	233

22.9	Destruktor	234
22.10	Datenkapselung, Datenabstraktion und Geheimnisprinzip	236
	22.10.1 Definitionen.....	236
	22.10.2 Zugriffsmethoden	238
	22.10.3 Properties.....	239
	22.10.4 Public-, Protected- und Private-Attribute.....	240
	22.10.5 Weitere Möglichkeiten der Properties	242
	22.10.6 Properties mit Dekoratoren	245
22.11	Stringausgaben mit str und repr	246
22.12	Klassenattribute	251
22.13	Statische Methoden	253
	22.13.1 Einleitendes Beispiel.....	254
	22.13.2 Getter und Setter für private Klassenattribute	255
22.14	Public-Attribute statt private Attribute	256
22.15	Magische Methoden und Operatorüberladung.....	258
	22.15.1 Einführung	258
	22.15.2 Übersicht magische Methoden.....	259
	22.15.3 Beispielklasse: Length	261
23	Bruchklasse	265
23.1	Brüche à la 1001 Nacht	265
23.2	Zurück in die Gegenwart.....	266
23.3	Rechenregeln	268
	23.3.1 Multiplikation von Brüchen	268
	23.3.2 Division von Brüchen.....	269
	23.3.3 Addition von Brüchen	270
	23.3.4 Subtraktion von Brüchen.....	270
	23.3.5 Vergleichsoperatoren	270
23.4	Integer plus Bruch	271
	23.4.1 Die Bruchklasse im Überblick	272
23.5	Fraction-Klasse.....	274
24	Aufrufbare Objekte	275
24.1	Einführung.....	275
	24.1.1 Die callable-Funktion.....	275
	24.1.2 Klassen statt Funktionen	276

25	Vererbung	279
25.1	Oberbegriffe und Oberklassen	279
25.2	Ein einfaches Beispiel	280
25.3	Überladen, Überschreiben und Polymorphie	281
25.4	Vererbung in Python.....	284
25.5	Klassenmethoden.....	287
25.6	Standardklassen als Basisklassen	289
26	Mehrfachvererbung	291
26.1	Einführung.....	291
26.2	Beispiel: KalenderUndUhr.....	292
26.3	Diamand-Problem oder „deadly diamond of death“	300
26.4	super und MRO	302
26.4.1	Ein einfaches Beispiel.....	302
26.4.2	Ein umfangreicheres Beispiel.....	305
27	Slots	307
27.1	Erzeugung von dynamischen Attributen verhindern	307
28	Dynamische Erzeugung von Klassen	309
28.1	Beziehung zwischen „class“ und „type“	309
29	Metaklassen	313
29.1	Motivation	313
29.2	Definition	318
29.3	Definition von Metaklassen in Python	318
29.4	Singletons mit Metaklassen erstellen	321
29.5	Beispiel: Methodenaufrufe zählen.....	322
29.5.1	Einführung	322
29.5.2	Vorbereitungen	322
29.5.3	Ein Dekorateur, um Funktionsaufrufe zu zählen	323
29.5.4	Die Metaklasse „Aufrufzähler“	324
30	Abstrakte Klassen	327
31	Aufgaben zur Objektorientierung	331

Teil IV: Funktionale Programmierung	335
32 Begriffsbestimmung	337
33 lambda, map, filter und reduce	339
33.1 lambda	339
33.2 map	341
33.3 Filtern von sequentiellen Datentypen mittels „filter“	344
33.4 reduce	344
33.5 Aufgaben	346
34 Listen-Abstraktion/List Comprehension	347
34.1 Die Alternative zu Lambda und Co.	347
34.2 Syntax	348
34.3 Weitere Beispiele	348
34.4 Die zugrunde liegende Idee	349
34.5 Anspruchsvolleres Beispiel	349
34.6 Mengen-Abstraktion	350
34.7 Rekursive Primzahlberechnung	351
34.8 Generatoren-Abstraktion	351
34.9 Aufgaben	352
35 Generatoren und Iteratoren	353
35.1 Einführung	353
35.2 Iteration in for-Schleifen	353
35.3 Generatoren	355
35.4 Endlos-Generatoren zähmen mit firstn und islice	359
35.5 Sinnvollere Beispiele	360
35.6 Beispiele aus der Kombinatorik	361
35.6.1 Permutationen	361
35.6.2 Variationen und Kombinationen	362
35.7 Generator-Ausdrücke	364
35.8 return-Anweisungen in Generatoren	365
35.9 send-Methode	366
35.10 Die close-Methode	370
35.11 Die throw-Methode	371
35.12 Dekoration von Generatoren	375
35.13 yield from	376
35.14 Aufgaben	378

36	Dekorateur	381
36.1	Einführung Dekorateur	381
36.1.1	Verschachtelte Funktionen	382
36.1.2	Funktionen als Parameter	384
36.1.3	Funktionen als Rückgabewert	385
36.1.4	Fabrikfunktionen	386
36.2	Ein einfacher Dekorateur	388
36.3	@-Syntax für Dekorateur	389
36.4	Anwendungsfälle für Dekorateur	392
36.4.1	Überprüfung von Argumenten durch Dekorateur	392
36.4.2	Funktionsaufrufe mit einem Dekorateur zählen	393
36.5	Dekorateur mit Parametern	395
36.6	Benutzung von Wraps aus functools	396
36.7	Eine Klasse als Dekorateur benutzen	398
36.8	Memoisation	399
36.8.1	Bedeutung und Herkunft des Begriffs	399
36.8.2	Memoisation mit Dekorateurfunktionen	399
36.8.3	Memoisation mit einer Klasse	400
36.8.4	Memoisation mit functools.lru_cache	401
Teil V: Weiterführende Themen		405
37	Tests und Fehler	407
37.1	Einführung	407
37.2	Modultests	409
37.3	Modultests unter Benutzung von <code>__name__</code>	410
37.4	doctest-Modul	412
37.5	Testgetriebene Entwicklung oder „Im Anfang war der Test“	415
37.6	unittest	417
37.7	Methoden der Klasse TestCase	419
37.8	Aufgaben	420
38	Daten konservieren	423
38.1	Persistente Speicherung	423
38.2	Pickle-Modul	424
38.2.1	Daten „einpökeln“ mit <code>pickle.dump</code>	424
38.2.2	<code>pickle.load</code>	425
38.3	Ein persistentes Dictionary mit <code>shelve</code>	425

39	Reguläre Ausdrücke	429
39.1	Ursprünge und Verbreitung	429
39.2	Stringvergleiche	429
39.3	Überlappungen und Teilstrings	431
39.4	Das re-Modul	431
39.5	Matching-Problem	432
39.6	Syntax der regulären Ausdrücke	434
39.6.1	Beliebiges Zeichen	434
39.7	Zeichenauswahl	435
39.8	Endliche Automaten	436
39.9	Anfang und Ende eines Strings	436
39.10	Vordefinierte Zeichenklassen	438
39.11	Optionale Teile	440
39.12	Quantoren	441
39.13	Gruppierungen und Rückwärtsreferenzen	443
39.13.1	Match-Objekte	443
39.14	Iteration über Matches mit finditer	446
39.15	Umfangreiche Übung	446
39.16	Alles finden mit findall	448
39.17	Alternativen	449
39.18	Compilierung von regulären Ausdrücken	450
39.19	Aufspalten eines Strings mit oder ohne regulären Ausdruck	450
39.19.1	split-Methode der String-Klasse	450
39.19.2	split-Methode des re-Moduls	452
39.19.3	Wörter filtern	454
39.20	Suchen und Ersetzen mit sub	455
39.21	Aufgaben	455
40	Typanmerkungen	459
40.1	Einführung	459
40.2	Einfaches Beispiel	460
40.3	Variablenanmerkungen	461
40.4	Listenbeispiele	462
40.5	Listen mit homogenem Typ	463
40.5.1	Version 3.6 bis 3.8	463
40.5.2	Python 3.9 und später	464

41	Systemprogrammierung	467
41.1	Einleitung	467
41.2	Häufig falsch verstanden: Shell	467
41.3	os-Modul	468
41.3.1	Vorbemerkungen	468
41.3.2	Umgebungsvariablen	469
41.3.3	Dateiverarbeitung auf niedrigerer Ebene	471
41.3.4	Weitere Funktionen im Überblick	476
41.3.5	os.path – Arbeiten mit Pfaden	490
41.4	shutil-Modul	498
41.5	glob-Modul	503
42	Forks	505
42.1	Fork	505
42.2	Fork in Python	505
Teil VI: Lösungen zu den Aufgaben		509
43	Lösungen zu den Aufgaben	511
43.1	Lösungen zu Kapitel 5 (Sequentielle Datentypen)	511
43.2	Lösungen zu Kapitel 6 (Listen und Tupel im Detail)	514
43.3	Lösungen zu Kapitel 7 (Verzweigungen)	516
43.4	Lösungen zu Kapitel 8 (Schleifen)	518
43.5	Lösungen zu Kapitel 9 (Dictionaries)	521
43.6	Lösungen zu Kapitel 12 (Dateien lesen und schreiben)	523
43.7	Lösungen zu Kapitel 15 (Funktionen)	524
43.8	Lösungen zu Kapitel 17 (Rekursive Funktionen)	529
43.9	Lösungen zu Kapitel 18 (Sortieren)	534
43.10	Lösungen zu Kapitel 20 (Alles über Strings ...)	537
43.11	Lösungen zu den Kapiteln 22 bis 31 (Aufgaben zur Objektorientierung)	540
43.12	Lösungen zu Kapitel 33 (lambda, map, filter und reduce)	554
43.13	Lösungen zu Kapitel 34 (Listen-Abstraktion/List Comprehension)	555
43.14	Lösungen zu Kapitel 35 (Generatoren und Iteratoren)	556
43.15	Lösungen zu Kapitel 37 (Tests und Fehler)	560
43.16	Lösungen zu Kapitel 39 (Reguläre Ausdrücke)	560
Stichwortverzeichnis		567



Vorwort

Ist es wirklich so, dass Vorworte – ähnlich wie Bedienungsanleitungen – meistens nicht gelesen werden? Auch wenn dies sicherlich für viele zutreffen mag, so gibt es Situationen, in denen gerade ein Vorwort wertvolle Dienste leisten kann. Zum Beispiel, um die Kaufentscheidung für ein Buch zu erleichtern. So stehen auch Sie jetzt vielleicht am Regal einer guten Buchhandlung und werden möglicherweise von zwei Fragen bewegt: Sie brauchen noch eine Bestätigung, dass Python die richtige Programmiersprache für Sie ist, und möchten wissen, wie Ihnen dieses Buch helfen wird, die Sprache schnell und effizient zu erlernen.

Für Python spricht der traumhafte Anstieg seiner Bedeutung in Wissenschaft, Forschung und Wirtschaft in den letzten Jahren. Dies spiegelt sich auch in den Rankings von Programmiersprachen wider, die von verschiedenen Stellen durchgeführt werden. PYPL zählt wohl zu den bekanntesten und anerkanntesten unter diesen Webrankings. Python führt im Dezember 2020 mit 30,34 % den Index von PYPL an, gefolgt von Java mit nur 17,23 %.¹

Weitere wichtige Gründe, Python zu lernen, sind: Python ist mit seiner einfachen natürlichen Syntax sehr einfach zu lernen. Python läuft auf allen Plattformen und erfreut sich auch beim Raspberry Pi größter Beliebtheit. Außerdem ist Python eine universell einsetzbare Programmiersprache, die sich bei den Aufgaben der Systemadministration ebenso effektiv einsetzen lässt wie im Maschinenbau, der Linguistik, der Pharmaindustrie, in der Banken- und Finanzwelt, der Physik, der Psychologie und vielen anderen Bereichen. Weil bedeutende Firmen wie Google, Facebook, Instagram, Spotify, Quora, Netflix und Dropbox Python benutzen und unterstützen, wird Python auch kontinuierlich weiterentwickelt.

Dieses Buch erscheint nun in der vierten, völlig überarbeiteten Ausgabe. Es eignet sich ebenso für Programmieranfänger als auch für Umsteiger von anderen Programmiersprachen. Behandelt werden nicht nur alle grundlegenden Sprachelemente von Python, sondern auch weiterführende Themen wie Generatoren, Dekorateur, Systemprogrammierung, Threads, Forks, Ausnahmebehandlungen und Modultests. Auf über hundert Seiten wird anschaulich und mit zahlreichen Beispielen auf die vielfältigen Aspekte der Objektorientierung eingegangen.

Brigitte Bauer-Schiewek, Lektorin

¹ Im Vorwort zur 3. Auflage stand hier: Im renommierten PYPL-Index stand Python im Juli 2017 auf dem zweiten Platz hinter Java. Während Java im Vergleich zum Vorjahr jedoch -1,1 % Anteil verloren hatte, gewann Python +4,0 % hinzu.

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, wie bereits bei allen vorigen Auflagen auch, 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 und www.python-course.eu danken, vor allem jenen, die sich mit konstruktiven Anmerkungen bei mir gemeldet haben. Wertvolle Anregungen erhielt ich auch von denen, die das Buch vorab Korrektur gelesen hatten: Stefan Günther für die Erstauflage des Buches. Für die Hilfe zur zweiten – weitestgehend überarbeiteten – Auflage möchte ich im besonderen Maße Herrn Jan Lendertse und Herrn Vincent Bermele Dank sagen. Für die dritte – wiederum stark veränderte – Auflage danke ich Herrn Wilhelm Wall für seine wertvolle Hilfe, insbesondere Tests unter macOS.

Mein besonderer Dank für die vierte Auflage gilt zum einen Herrn Tobias Habermann und zum anderen Herrn Dr. Konrad Wienands. Herr Habermann hat dafür gesorgt, dass alle Python-Beispiele dieser Auflage automatisch mittels „Pythontex“ getestet und ausgeführt werden. Herr Dr. Wienands ist mit großem Sachverstand in die Rolle eines potenziellen Anfängers geschlüpft und hat mir viele wertvolle Hinweise gegeben, wo es möglicherweise offene Fragen oder Probleme bei Neulingen geben könnte. Außerdem hat er auch kleine Unstimmigkeiten gefunden.

Vergessen darf ich auch nicht all diejenigen, die die Exemplare der vorigen Auflagen gekauft und damit erst den Erfolg des Buches ermöglicht hatten. Ebenso danke ich all diejenigen, die mich in E-Mails auf kleine Fehler oder Unstimmigkeiten hingewiesen haben, die ich aber hier leider nicht alle namentlich erwähnen kann.

Zuletzt danke ich auch ganz herzlich dem Hanser Verlag, der dieses Buch – nun auch in der vierten Auflage – ermöglicht. Vor allem danke ich Frau Brigitte Bauer-Schiewek und Frau Kristin Rothe, Programmplanung Computerbuch, für die kontinuierliche ausgezeichnete Unterstützung. Für die technische Unterstützung bei LaTeX-Problemen danke ich Herrn Stephan Korell und Frau Irene Weilhart. Herrn Jürgen Dubau danke ich fürs Lektorat.

Bernd Klein, Singen

Teil I

Einleitung



1

Einleitung

■ 1.1 Einfach und schnell zu lernen

Python ist eine Programmiersprache, die in vielerlei Hinsicht begeistert. Ein riesiger Vorteil im Vergleich zu anderen Sprachen liegt in der leichten Erlernbarkeit der Sprache. In der ersten Auflage führte das Buch den Untertitel „In einer Woche programmieren lernen“. Kann man wirklich in so kurzer Zeit programmieren lernen? Das ist möglich! Vor allem mit Python. Wir erfahren es mehrmals im Monat in unseren meist fünftägigen Python-Kursen. Sie werden zum einen von Leuten, die noch keinerlei Programmiererfahrung haben, und zum anderen von Programmierenden mit Erfahrungen in C, C++, Java, Perl und anderen Programmiersprachen besucht. Manche haben auch schon vorher Erfahrungen in Python gesammelt. Aber eines ist in allen Gruppen gleich: Wir haben es immer geschafft, dass jeder programmieren gelernt hat, und vor allen Dingen – was uns am wichtigsten ist – konnten wir immer die Begeisterung für die Sprache Python entfachen.

So ist es auch eines der wichtigsten Ziele dieses Buches, die Leserinnen und Leser möglichst schnell und mit viel Freude zum selbständigen Programmieren zu bringen. Komplexe Sachverhalte werden in einfachen, leicht verständlichen Diagrammen veranschaulicht und an kleinen Beispielen eingeübt, die sich im Laufe vieler Schulungen herausgebildet und bewährt haben.

■ 1.2 Geschichte von Python

Was hat Python mit dem lateinischen Alphabet zu tun? Beide beginnen mit ABC. Im Falle von Python ist es die Programmiersprache ABC. Die Sprache wurde Anfang der 1990er Jahre von Guido van Rossum am Zentrum für Mathematik (Centrum voor Wiskunde en Informatica) in Amsterdam entwickelt. Ursprünglich war Python als Nachfolger für die Lehrsprache ABC entwickelt worden und sollte auf dem verteilten Betriebssystem Amoeba laufen. Guido van Rossum hatte auch an der Entwicklung der Sprache ABC mitgewirkt. In einem Interview mit Bill Venners sagte er: „In den frühen 1980er Jahren arbeitete ich als Implementierer in einem Team, das eine Sprache namens ABC am Centrum voor Wiskunde en Informatica (CWI) aufbaute. Ich weiß nicht, wie gut die Leute den Einfluss von ABC auf Python kennen. Ich versuche, den Einfluss von ABC zu erwähnen, weil ich alles, was

ich während dieses Projekts gelernt habe, und die Leute, die daran gearbeitet haben, zu verdanken habe.”¹

Auch wenn sich das offizielle Logo von Python aus zwei abstrahierten Python-Schlangen zusammensetzt, hat der Name der Programmiersprache Python herkunftsmäßig nichts mit Schlangen zu tun. Für Guido van Rossum stand vielmehr die britische Komikertruppe „Monty Python” mit ihrem legendären „Flying Circus” Pate für den Namen. Guido van Rossum schrieb dazu: „Vor über sechs Jahren, im Dezember 1989, suchte ich nach einem ‚Hobby’-Programmierprojekt, mit dem ich mich während der Weihnachtswoche beschäftigen könnte. Mein Büro (ein staatliches Forschungslabor in Amsterdam) würde geschlossen sein, aber ich hatte einen Computer und nicht viel anderes vor. Ich beschloss, einen Interpreter für die neue Skriptsprache zu schreiben, über die ich in letzter Zeit nachgedacht hatte: einen Nachkommen von ABC, der Unix-/C-Hacker ansprechen würde. Ich habe Python als Arbeitstitel für das Projekt gewählt, da ich in einer etwas respektlosen Stimmung bin (und ein großer Fan von ‚Monty Python’s Flying Circus’).”²

Dennoch sind Assoziationen mit Schlangen möglich und sinnvoll: Man denke nur an das Python-Toolkit „Boa” oder die Programmiersprache Cobra. Außerdem ist eine Schlange, wie schon erwähnt, im Python-Logo.

■ 1.3 Zen von Python

Vom Softwareentwickler Tim Peters stammt das „Zen of Python”, welches er 1999 veröffentlichte. Dabei handelt es sich um „Leitprinzipien” zum Schreiben guter Python-Programme. Prinzipien, die sich sowohl an Programmierende als auch an die Designer der Sprache richten und die Sprache Python auch entscheidend prägten:

- Schön ist besser als hässlich.
- Explizit ist besser als implizit.
- Einfach ist besser als komplex.
- Komplex ist besser als kompliziert.
- Flach ist besser als verschachtelt.
- Spärlich ist besser als dicht.
- Lesbarkeit zählt.
- Sonderfälle sind nicht speziell genug, um gegen die Regeln zu verstoßen.
- Obwohl Praktikabilität die Reinheit übertrifft,
- Fehler sollten niemals stillschweigend ablaufen.
- Sofern nicht ausdrücklich zum Schweigen gebracht.
- Widerstehe im Angesicht von Zweideutigkeiten der Versuchung zu raten.

¹ Bill Venners: The Making of Python, A Conversation with Guido van Rossum, Part I, January 13, 2003, <https://www.artima.com/intv/python.html>

² Guido van Rossum schrieb dies in einem Vorwort zur ersten Auflage des Buches „Programming Python” von Marc Lutz.

- Es sollte einen – und vorzugsweise nur einen – offensichtlichen Weg geben, es zu tun.
- Obwohl dieser Weg zunächst vielleicht nicht offensichtlich ist, es sei denn, Sie sind Niederländer.
- Jetzt ist besser als nie.
- Obwohl nie oft besser ist als gerade jetzt.
- Wenn die Implementierung schwer zu erklären ist, ist es eine schlechte Idee.
- Wenn die Implementierung leicht zu erklären ist, kann dies eine gute Idee sein.
- Namespaces sind eine großartige Idee – lasst uns mehr davon machen!

■ 1.4 Zielgruppe des Buches

Beim Schreiben eines Buches stellt man sich ein Gegenüber vor, eine Person, die das Geschriebene lesen, mögen und verstehen soll. Diese Person kann ein beliebiges Geschlecht haben, deswegen haben wir uns im Buch um geschlechtsneutrale Formulierungen bemüht. Begriffe wie „der Nutzer“ oder im Plural „die Nutzer“ kommen nicht mehr im Buch vor. Allerdings haben wir diese auch nicht durch Formulierungen wie „die NutzerInnen“ bzw. „Nutzer*innen“ ersetzt, weil diese Formulierungen von einigen abgelehnt werden. Wir denken, dass es uns gelungen ist, geschlechtsneutral zu formulieren, ohne dass der Lese-
fluss gestört wird oder die Gefühle von manchen verletzt werden.

Natürlich hatten wir beim Schreiben aber nicht nur eine Person im Blickfeld, sondern eine ganze Schar von Lesenden. Da sind zum einen diejenigen, die noch nie programmiert haben und Sachverhalte erklärt haben wollen, die Leute mit Programmiererfahrung in anderen Sprachen vielleicht als „trivial“ oder „selbstverständlich“ bezeichnen würden. Aber hier ist ein Buch wohl dem Präsenzunterricht, also wenn Lehrende und Lernende am gleichen Ort zusammen sind, deutlich überlegen: Ist einem der Stoff eines Abschnitts oder sogar Kapitels bereits vertraut, kann man es einfach überspringen, bevor man sich zu langweilen beginnt. Ebenso können, falls keine Kenntnisse in anderen Programmiersprachen vorhanden sind, Abschnitte übersprungen werden, in denen wir Ähnlichkeiten, aber auch generelle Unterschiede in der Vorgehensweise von Python im Vergleich zu anderen Sprachen herausarbeiten.

In der obigen Aufzählung fehlt aber noch eine wichtige Gruppe, nämlich diejenigen, die schon Erfahrungen mit Python haben. Dies ist eine Gruppe mit einer breiten Streuung: angefangen bei denjenigen, die bereits ein wenig reingeschnuppert haben, gefolgt von solchen, die bereits kleine oder auch größere Programme geschrieben haben, bis hin zu jenen, die sich als Experten bezeichnen. Unsere Erfahrungen zeigen, dass auch einige aus der zuletzt genannten Gruppe neue Einblicke und Sachverhalte in diesem Buch finden werden, die sich positiv auf deren zukünftige Programmierung mit Python auswirken.

Ansonsten kann dieses Buch auch bestens als Nachschlagewerk benutzt werden. Der umfangreiche Index in diesem Buch macht das Auffinden besonders einfach und erlaubt es damit, dieses Buch außerdem als Referenz zu verwenden, auch wenn es keinesfalls unter diesem Aspekt geschrieben worden ist.

■ 1.5 Aufbau des Buches

Dieses Buch besteht aus sechs Teilen:

- Teil I: Wir haben uns entschieden, die Einleitung, in der Sie sich gerade befinden, als eigenen Teil des Buches aufzunehmen. Schließlich enthält sie wichtige Informationen und sollte nach Möglichkeit auch gelesen werden.
- Teil II: Im zweiten Teil behandeln wir die Grundlagen der Sprache. Wir lernen Variablen und deren Besonderheit in Python kennen. In diesem Zusammenhang lernen wir auch die Datenstrukturen wie ganze Zahlen (Integers), Fließkommazahlen (Floats), Zeichenketten (Strings), Listen, Tupel, Dictionaries und Mengen kennen. Wir beschäftigen uns eingehend mit allen Anweisungsarten von Python, d.h. Zuweisungen, Kontrollstrukturen wie bedingten Anweisungen und Schleifen sowie Möglichkeiten der Ein- und Ausgabe. Außerdem behandeln wir eingehend Funktionen, Definitionen, einfache und auch komplexere Beispiele, Parameterübergabe sowie Gültigkeitsbereiche von Variablen, und in einem umfangreichen Kapitel gehen wir auch auf rekursive Funktionen ein. Die Modularisierung in Modulen und Paketen ist ein weiterer Themenschwerpunkt.
- Teil III: Den dritten Teil des Buches haben wir – entsprechend ihrer Bedeutung – ganz der Objektorientierung gewidmet. In anschaulichen Beispielen führen wir sanft in das objektorientierte Denken und Programmieren ein und zeigen die Besonderheiten von Python deutlich auf. Danach fahren wir mit den fortgeschrittenen Themen der Objektorientierung fort und behandeln neben Vererbung, Mehrfachvererbung und Slots auch extrem fortgeschrittene Themen wie Metaklassen und abstrakte Klassen.
- Teil IV: Viele lieben Python, weil es neben der objektorientierten Programmierung auch wesentliche Konzepte der funktionalen Programmierung unterstützt. Deshalb widmen wir diesem Thema den vierten Hauptteil des Buches. Dieser Teil des Buches wendet sich eher an Personen mit Programmiererfahrung, aber es lohnt, sich die behandelten Konzepte anzueignen. Sie sind notwendig, um Python richtig zu verstehen. Die Listen-Abstraktionen bieten ebenso wie die `lambda`-, `map`-, `filter`- und `reduce`-Funktionen eine faszinierende Möglichkeit, die Programmierung auf ein abstrakteres Level zu bringen. Dadurch kann man komplexere Probleme mit geringerem Programmieraufwand lösen, was außerdem zu einer besseren Verständlichkeit und einfacheren Wartbarkeit der Programme führt. Dennoch kann man die gleichen Programme auch ohne diese Techniken schreiben. Eingehend befassen wir uns auch mit dem Themenkomplex „Generatoren und Iteratoren“ sowie „Dekoration und Dekorateure“!
- Teil V: Im fünften Teil mit dem Titel „[Weiterführende Themen](#)“ verlassen wir das „eigentliche“ Python und wenden uns weiterführenden Themen der Programmierung zu. Diesen Teil hätten wir auch mit „Anwendungsprogrammierung“ überschreiben können. Testverfahren und Debugging gehört zu den Fragestellungen, die alle Programmierinnen und Programmierer bewegen sollten. Im Prinzip hätten wir dieses Kapitel ebenso gut in den ersten Teil nehmen können. Außerdem behandeln wir in einem Kapitel „Reguläre Ausdrücke“, die nahezu unerlässlich sind, wenn man Textverarbeitung effizient betreiben will. Man braucht sie beispiels-

weise, wenn man aus Log- oder Parameterdateien bestimmte Informationen herausfiltern will. Den ab Python 3.5 eingeführten Typenmerkungen (*type annotations*) haben wir auch ein eigenes Kapitel gewidmet. In einem weiteren Kapitel zeigen wir außerdem, wie man Daten strukturerhaltend übers Programmieren hinaus konservieren kann. Das Kapitel zur Systemprogrammierung dürfte von besonderer Bedeutung für Leute sein, die gerne mit dem Betriebssystem agieren möchten, damit sie ihre Systemprogramme zukünftig unter Python und nicht mehr mit Shell-Skripting verfassen können.

Teil VI: Programmieren lernen ist vor allen Dingen eine aktive Tätigkeit. Nur ein Buch zu lesen und Beispiele nachzuvollziehen, genügt nicht. Deshalb finden Sie zu den meisten Kapiteln interessante und lehrreiche Übungsaufgaben, die den Stoff vertiefen. Aufgaben ohne Lösungen wären aber wenig sinnvoll, weshalb sich im letzten Teil dieses Buches ausführliche Musterlösungen mit Erläuterungen zu den Aufgaben befinden.

Teil VII: Auch wenn es nicht als eigener Teil aufgeführt ist, wollen wir hier noch auf das Stichwortverzeichnis eingehen. Erst ein umfangreiches Stichwortverzeichnis macht aus einem guten Buch ein nützliches und brauchbares Buch!

■ 1.6 Programmieren lernen „interaktiv“

Wie bereits erwähnt, floss in dieses Buch die jahrelange Erfahrung sowohl in der Theorie und Praxis des Programmierens allgemein, aber vor allem auch die Vermittlung des Stoffes in zahlreichen kleinen und großen Kursen mit unterschiedlichsten Besuchertypen ein. Aber ein Buch zu schreiben, stellt dennoch eine neue Herausforderung dar. Beim Buch fehlt leider die direkte Interaktion zwischen dem, der das Wissen vermittelt, und dem Lernenden. Vor Ort kann man sofort sehen, wenn sich bei einem Teilnehmer die Stirn runzelt, große Fragezeichen erscheinen oder wenn sich jemand aus Stress das Ohr läppchen zu reiben beginnt. Dann weiß man als erfahrener Dozent, dass es höchste Zeit ist, ein paar zusätzliche Beispiele und Analogien zu verwenden, oder dass man den Stoff nochmals in anderen – möglicherweise auch einfacheren – Worten erklären sollte. Beim Schreiben eines Buches muss man diese möglichen Klippen vorhersehen und die nötigen zusätzlichen Übungen und Beispiele an den entsprechenden Stellen bereitstellen. Aber was bei vielen Lesern hilft, diese Klippen zu umgehen, führt bei anderen nun vielleicht zu Langeweile und Ungeduld, denn sie empfinden diese zusätzlichen Erklärungen, Übungen oder Beispiele möglicherweise als Zeit- oder Platzvergeudung.

Das Grundproblem ist, dass es sich bei einem Buch nicht um ein interaktives Medium handelt. Aber dank des Internets können wir Ihnen diese Interaktivität dennoch bieten. Im nächsten Abschnitt finden Sie die Adressen, wo Sie Hilfe und zusätzliche Informationen zum Buch finden.

■ 1.7 Download der Beispiele und Hilfe

Alle im Buch verwendeten Beispiele finden Sie zum Download unter

<http://www.python-kurs.eu/buch/beispiele/>

Auch wenn wir das Buch so geschrieben haben, dass Sie ohne zusätzliche Hilfe auskommen sollten, wird es dennoch hier und da mal ein Problem geben, wo Sie sich vielleicht festgebissen haben. Wenn das Glück es so will, dass Sie in einer Umgebung arbeiten, in der es andere Python-Programmierende gibt, haben Sie es natürlich gut. Aber viele Leserinnen und Leser genießen diesen Vorteil nicht. Dann könnte sich ein Besuch unserer Website besonders lohnen:

<http://www.python-kurs.eu/buch/>

Dort finden Sie ein Korrekturverzeichnis, zusätzliche bzw. aktualisierte Übungen und sonstige Hilfestellungen. Ansonsten bleibt natürlich immer noch der Einsatz einer Suchmaschine!

■ 1.8 Anregungen und Kritik

Falls Sie glauben, eine Ungenauigkeit oder einen Fehler im Buch gefunden zu haben, können Sie auch gerne eine E-Mail direkt an den Autor schicken: klein@python-kurs.eu.

Natürlich gilt dies auch, wenn Sie Anregungen oder Wünsche zum Buch geben wollen. Leider können wir jedoch – so gerne wir es auch tun würden – keine individuellen Hilfen zu speziellen Problemen geben.

Wir werden versuchen, Fehler und Anmerkungen in kommenden Auflagen zu berücksichtigen. Selbstverständlich aktualisieren wir damit auch unsere Informationen unter

<http://www.python-kurs.eu/buch/>