



Ralph Steyer

Programmierung in Python

Ein kompakter Einstieg für die Praxis

EBOOK INSIDE

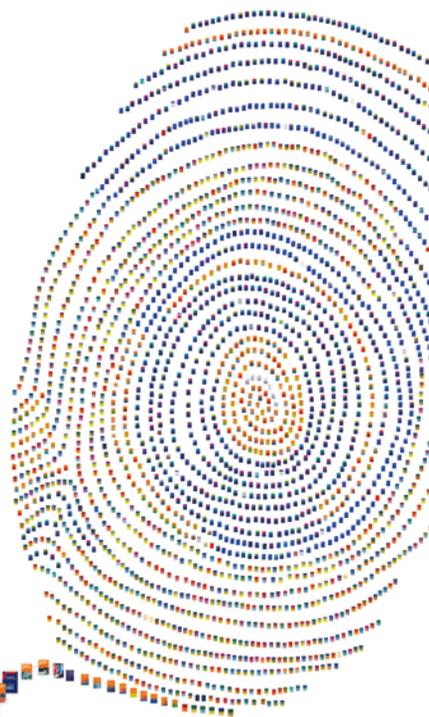
 Springer Vieweg

Programmierung in Python

Lizenz zum Wissen.

Sichern Sie sich umfassendes Technikwissen mit Sofortzugriff auf tausende Fachbücher und Fachzeitschriften aus den Bereichen: Automobiltechnik, Maschinenbau, Energie + Umwelt, E-Technik, Informatik + IT und Bauwesen.

Exklusiv für Leser von Springer-Fachbüchern: Testen Sie Springer für Professionals 30 Tage unverbindlich. Nutzen Sie dazu im Bestellverlauf Ihren persönlichen Aktionscode **C0005406** auf www.springerprofessional.de/buchaktion/



Jetzt
30 Tage
testen!

Springer für Professionals.
Digitale Fachbibliothek. Themen-Scout. Knowledge-Manager.

-  Zugriff auf tausende von Fachbüchern und Fachzeitschriften
-  Selektion, Komprimierung und Verknüpfung relevanter Themen durch Fachredaktionen
-  Tools zur persönlichen Wissensorganisation und Vernetzung

www.entschieden-intelligenter.de

Springer für Professionals

 Springer

Ralph Steyer

Programmierung in Python

Ein kompakter Einstieg für die Praxis

Ralph Steyer
Bodenheim, Deutschland

ISBN 978-3-658-20704-5 ISBN 978-3-658-20705-2 (eBook)
<https://doi.org/10.1007/978-3-658-20705-2>

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

Springer Vieweg

© Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt 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.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen. Der Verlag bleibt im Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutionsadressen neutral.

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer Fachmedien Wiesbaden GmbH und ist Teil von Springer Nature

Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Vorwort

„And now for something completely different.“ Was ist der Sinn des Lebens? Wie geht es Brian? Was machen Ritter mit den Kokosnüssen? Ist die Schwerkraft wirklich so wunderbar? Was macht man mit diesem lebensgefährlichen Witz? Und was haben diese Fragen mit dem Buch zu tun? Fragen Sie für die Antworten einfach einmal den Erfinder von Python – Guido van Rossum. Denn der war bei der Entwicklung der Sprache ein großer Fan der englischen Komikertruppe Monty Python und deren Show Monty Python’s Flying Circus. Und nun fragen Sie sich immer noch, woher der Name „Python“ stammt? Ich überlasse die Antwort Ihrer Phantasie oder intensiven Recherchen im Internet. Oder Ihrer Aufmerksamkeit beim Lesen dieses Buchs, denn ich nehme mir die Freiheit und lasse immer wieder Zitate von Monty Python einfließen – denn auch ich bin Fan der Truppe.

Die Programmiersprache Python hat sich in den vergangenen Jahren zu einem Schwerkgewicht in der Programmiererszene entwickelt. Offensichtlich trifft das Konzept von Python den Nerv der Zeit. Oder genauer gesagt: Das Konzept stellt Ansätze, Lösungen und Vorgehensweisen für Probleme bereit, die andere Sprachen so nicht bieten und viele Leute interessant finden.

Python gilt aktuell als eine der beliebtesten Einsteigersprachen überhaupt, denn Python wurde mit dem Ziel größter Einfachheit und Übersichtlichkeit entworfen. Zentrales Ziel bei der Entwicklung der Sprache war die Förderung eines gut lesbaren, knappen Programmierstils. In vielen Ländern hat Python an Universitäten bei Anfängerkursen in informatikbezogenen Studiengängen Java abgelöst, was über viele Jahre die Szene beherrscht hatte und im professionellen Umfeld immer noch das Maß aller Dinge darstellt. Wobei man erwähnen muss, dass viele neue Python-Programmierer dann später zusätzlich Java oder eine andere OO-Sprache wie C# lernen oder auch darauf umsteigen. Aber auch für diesen Weg legt Python mit dem Zwang zu einem strukturierten, klaren Programmierstil eine hervorragende Grundlage – wenn man dessen Freiheiten nicht missbraucht.

Umgekehrt lässt sich Python sehr schnell erfassen, wenn man bereits Erfahrung mit anderen, weitgehend beliebigen Programmiersprachen hat. Denn Python unterstützt sowohl die objektorientierte, die aspektorientierte, die strukturierte als auch die funktionale Programmierung. Das bedeutet, Python zwingt den Programmierer nicht zu einem

einzigem Programmierstil, sondern erlaubt, das für die jeweilige Aufgabe am besten geeignete Paradigma zu wählen. Und damit können ebenso Erfahrungen aus anderen Programmierkonzepten mehr oder weniger direkt weitergenutzt werden. Dieser universell mögliche Zugang ist neben der Einfachheit vermutlich eines der Erfolgsrezepte von Python.

Nun noch kurz zu meiner Person. Ich habe vor vielen Jahren in Frankfurt/Main an der Goethe-Universität Mathematik studiert (Diplom) und danach anfangs einen recht typischen Werdegang für Mathematiker genommen: Ich bin bei einer großen Versicherung gelandet – aber schon da mit IT-Schwerpunkt. Zuerst habe ich einige Jahre als Programmierer mit Turbo Pascal und später mit C und C++ gearbeitet. Nach vier Jahren habe ich in die fachliche Konzeption für eine Großrechnerdatenbank unter MVS gewechselt. Die Erfahrung war für meinen Schritt in die Selbständigkeit sehr motivationsfördernd, denn mir wurde klar, dass ich das nicht auf Dauer machen wollte. Seit 1996 verdiene ich daher meinen Lebensunterhalt als Freelancer, wobei ich fliegend zwischen der Arbeit als Fachautor, Fachjournalist, EDV-Dozent, Consultant und Programmierer wechsele. Daneben referiere ich gelegentlich auf Web-Kongressen, unterrichte an verschiedenen Akademien und Fachhochschulen oder nehme Videotrainings auf. Das macht aus meiner Sicht einen guten Mix aus, bewahrt vor beruflicher Langeweile und hält mich in der Praxis als auch am Puls der Entwicklung. Insbesondere habe ich das Vergnügen als auch die Last, mich permanent über neue Entwicklungen auf dem Laufenden zu halten, denn die Halbwertszeit von Computerwissen ist ziemlich kurz. Dementsprechend ist mein Job zwar anstrengend, aber vor allem immer wieder spannend. Wenn Sie weitere und detaillierte Informationen zu meiner Person erhalten wollen, finden Sie mich natürlich im Internet. Ich bin in diversen sozialen Netzwerken und natürlich auch mit einer eigenen Webseite im weltweiten Web unterwegs. Etwa hier:

<http://www.rjs.de>

<http://blog.rjs.de>

Doch lassen Sie uns also jetzt mit Python beginnen.

Ihr Autor

Ralph Steyer

Herbst/Winter 2017/2018

Inhaltsverzeichnis

1	Einleitung und Grundlagen – Bevor es richtig losgeht	1
1.1	Was behandeln wir in dem einleitenden Kapitel?	1
1.2	Das Ziel des Buchs.	1
1.3	Was sollten Sie bereits können?.	2
1.4	Was ist Python?	2
1.4.1	Das Ziel von Python.	3
1.4.2	Was umfasst Python?	3
1.4.3	Die verschiedenen Python-Paradigma	4
1.5	Was benötigen Sie zum Arbeiten mit dem Buch?	4
1.5.1	Hardware und Betriebssystem	4
1.5.2	Die Python-Version	4
1.5.3	Python laden und installieren.	5
2	Erste Beispiele – Der Sprung ins kalte Wasser.	19
2.1	Was behandeln wir in diesem Kapitel?	19
2.2	Der Interaktivmodus – die Kommandozeile von Python.	19
2.2.1	Das Prompt.	21
2.2.2	Der Hilfemodus in der Kommandozeile	25
2.3	Anweisungen in (echten) Quelltext auslagern	27
2.4	IDLE & Co.	28
2.4.1	Weitere IDEs und Editoren für Python	31
3	Built-in-Functions – Modularisierung durch Unterprogramme	33
3.1	Was behandeln wir in diesem Kapitel?	33
3.2	Was sind Funktionen im Allgemeinen?	33
3.3	Built-in Functions	34
3.3.1	Hilfe zu Built-in-Functions im Hilfemodus.	34
3.3.2	Hilfe zu Built-in-Functions im Editormodus.	35
3.3.3	Die print()-Funktion.	36
3.3.4	Die input()-Funktion	41
3.3.5	Eine kurze Übersicht aller Built-in Functions	42

4	Grundlegende Begriffe – Kommentare, SheBang und Strukturanalysen . . .	47
4.1	Was behandeln wir in diesem Kapitel?	47
4.2	Token und Parser	47
4.2.1	Zerlegen von Quelltext.	48
4.3	Kommentare.	49
4.3.1	Kommentare in Python	50
4.4	SheBang und eine Python-Datei direkt ausführen.	52
4.4.1	SheBang als besonderer Kommentar	52
5	Anweisungen – Dem Computer Befehle geben	53
5.1	Was behandeln wir in diesem Kapitel?	53
5.2	Was sind Anweisungen?.	53
5.2.1	Eine Frage der Reihenfolge	53
5.3	Anweisungsarten	54
5.3.1	Blockanweisung	54
5.3.2	Kontrollflussanweisungen	55
5.3.3	Deklarationsanweisung	55
5.3.4	Ausdrucksanweisung	55
5.3.5	Die leere Anweisung <i>pass</i>	56
6	Datentypen, Variablen und Literale – Die Art der Information	57
6.1	Was behandeln wir in diesem Kapitel?	57
6.2	Variablen	57
6.2.1	Variablen deklarieren	57
6.2.2	Variablen im Quellcode verwenden	59
6.3	Die Datentypen in Python	59
6.3.1	Lose Typisierung und Typumwandlung in Python	59
6.3.2	Die Python-Datentypen	61
6.3.3	Zahlen – <i>int</i> , <i>float</i> und <i>complex</i>	63
6.3.4	Zeichenliterale	67
6.4	Den Datentyp bestimmen und umwandeln	68
6.4.1	Den Datentyp mit <i>type()</i> dynamisch bestimmen	68
6.4.2	Implizite und explizite Typumwandlung	69
7	Ausdrücke, Operatoren und Operanden – Die Verarbeitung von Daten . . .	71
7.1	Was behandeln wir in diesem Kapitel?	71
7.2	Ausdrücke	71
7.3	Operationen mit Operatoren und Operanden.	72
7.3.1	Arithmetische Operatoren	72
7.3.2	Der String-Verkettungsoperator.	75
7.3.3	Zuweisungsoperatoren	75
7.3.4	Boolesche Operatoren (Vergleichsoperatoren)	77
7.3.5	Logische Operatoren	78

7.3.6	Die Membership-Operatoren	79
7.3.7	Identitätsoperatoren	80
7.3.8	Bitweise Operatoren.	80
7.4	Operatorvorrang und Ausdrucksbewertung	85
7.4.1	Die Priorität der Python-Operatoren	85
7.4.2	Bewertung von Ausdrücken	85
8	Kontrollstrukturen – Die Steuerung des Programmflusses	87
8.1	Was behandeln wir in diesem Kapitel?	87
8.2	Was sind Kontrollstrukturen?	87
8.3	Die Kontrollstrukturen in Python.	88
8.3.1	Entscheidungsanweisungen	88
8.3.2	Iterationsanweisungen	94
8.3.3	Sprunganweisungen	96
9	Funktionen in Python – Modularisierung mit „Unterprogrammen“	99
9.1	Was behandeln wir in diesem Kapitel?	99
9.2	In Python eigene Funktionen deklarieren – das Schlüsselwort def	99
9.2.1	Übergabewerte	100
9.2.2	Rückgabewerte.	101
9.3	Funktionen aufrufen.	101
9.3.1	Stehen in Python global deklarierte Variablen in der Funktion zur Verfügung?	103
9.4	Rekursion	104
9.5	Innere Funktionen – Closures	107
9.6	Lambda-Ausdrücke und anonyme Funktionen	108
9.6.1	Lambda-Funktionen verwenden	109
9.7	Besondere Situationen bei Funktionen in Python	110
9.7.1	Lokale Variablen in Funktionen	110
9.7.2	Die Anzahl der Parameter passen nicht	111
9.7.3	Unerreichbarer Code	114
10	Sequenzielle Datenstrukturen – Mehrere Informationen gemeinsam verwalten	115
10.1	Was behandeln wir in diesem Kapitel?	115
10.2	Was sind sequenzielle Datenstrukturen?	115
10.2.1	Zeichenketten als sequenzielle Ansammlung von Zeichenliteralen	115
10.2.2	Arrays.	116
10.3	Tupel	116
10.3.1	Verschachtelte Tupel	117
10.3.2	Tupel und der Membership-Operator	118
10.3.3	Einzelne Einträge in Tupel ansprechen	120
10.3.4	Die Anzahl der Elemente in einem Tupel bestimmen	124

10.4	Dynamische Listen.	124
10.4.1	Warum Listen und Tupel?	125
10.5	Methoden für Listen.	125
10.5.1	Verschiedene Listenmethoden in einem Beispiel	126
10.5.2	Einen Stack erzeugen.	127
10.5.3	Eine Queue mit einer Liste erzeugen.	128
10.6	Dictionaries	129
10.6.1	Spezielle Methoden für Dictionaries	130
10.6.2	Ein Beispiel zum allgemeinen Umgang mit Dictionaries	131
10.6.3	Ein Dictionary aktualisieren oder erweitern	132
10.6.4	Iteration über ein Dictionary	133
10.7	Mengen	134
10.7.1	Vereinfachte Notation	134
10.7.2	Operationen auf „set“-Objekten.	135
10.8	Operatoren bei sequenziellen Datentypen	136
10.8.1	Der Plusoperator	137
10.8.2	Multiplikationen mit sequenziellen Datentypen	137
10.8.3	Inhalt überprüfen	138
10.9	Über sequenzielle Strukturen iterieren.	140

11 Objektorientierte Programmierung in Python – Klassen, Objekte, Eigenschaften und Methoden.

11.1	Was behandeln wir in diesem Kapitel?	143
11.2	Hintergründe der OOP	143
11.2.1	Ziele der OOP – Wiederverwendbarkeit und bessere Softwarequalität.	144
11.2.2	Kernkonzepte der Objektorientierung	145
11.3	Klassen.	146
11.3.1	Klassen als Baupläne, Konstruktoren und Destruktoren	147
11.3.2	Der konkrete Klassenaufbau in Python	147
11.3.3	Die konkrete Instanziierung.	148
11.4	Details zu Objekten	150
11.4.1	OO-Philosophie als Abstraktion	151
11.4.2	Instanzelemente versus Klasselemente	151
11.4.3	Der Aufbau von Objekten in Python	152
11.4.4	Zugriff auf Objektbestandteile.	152
11.4.5	Von Grund auf objektorientiert	157
11.5	Klassenmethoden und statische Methoden	157
11.5.1	Klassenmethoden.	158
11.5.2	Statische Methoden	159
11.6	Eine Frage der Sichtbarkeit	161
11.6.1	Ein Beispiel für den Zugriff auf ein öffentliches Element.	161

11.6.2	Ein Beispiel für den versuchten Zugriff auf ein privates Element von außen.	162
11.6.3	Getter und Setter	162
11.7	Ein Objekt löschen.	165
11.7.1	Ein Beispiel für das Redefinieren des Destruktors	165
11.8	Ein paar besondere OO-Techniken	166
11.8.1	Eine To-String-Funktionalität bereitstellen – <code>__str__</code>	166
11.8.2	Objekte dynamisch erweitern, das Dictionary <code>__dict__</code> und Slots	167
11.8.3	Dynamische Erzeugung von Klassen, Metaklassen und die Klasse <code>type</code>	169
11.9	Vererbung.	170
11.9.1	Grundlagentheorie zur Vererbung	170
11.9.2	Umsetzung von Vererbung in Python.	172
11.9.3	Mehrfachvererbung in Python	172
11.9.4	Polymorphie über Überschreiben und Überladen	173
11.10	Was ist mit Schnittstellen und abstrakten Klassen in Python?.	175
11.10.1	Abstrakte Superklassen	176
11.10.2	Was ist im Allgemeinen eine Schnittstelle?.	176
11.11	Module und Pakete.	177
11.11.1	Die <code>import</code> -Anweisung.	177
11.11.2	Importieren mit <code>from</code>	178
11.11.3	Pakete.	178
11.11.4	Das Python-API.	180
12	Exceptionhandling – Ausnahmsweise	181
12.1	Was behandeln wir in diesem Kapitel?	181
12.2	Was sind Ausnahmen?	181
12.3	Warum ein Ausnahmekonzept?	182
12.4	Konkrete Ausnahmebehandlung in Python	183
12.4.1	Ein erstes Beispiel mit einfacher Ausnahmebehandlung.	183
12.4.2	Mehrere Ausnahme-Blöcke	184
12.4.3	Die <code>finally</code> -Anweisung.	184
12.4.4	Praktische Beispiele.	185
12.5	Standard Exceptions.	188
12.5.1	Die Reihenfolge bei mehreren Ausnahmetypen	189
12.6	Der <code>else</code> -Block	189
12.7	Ausnahmeobjekte auswerten	190
12.8	Werfen von Ausnahmen mit <code>raise</code>	191
12.9	Eigene Ausnahmeklassen definieren	193
12.10	Die <code>assert</code> -Anweisung	193

13	String-Verarbeitung in Python – Programmierete Textverarbeitung	195
13.1	Was behandeln wir in diesem Kapitel?	195
13.2	Typische String-Verarbeitungstechniken	196
13.3	Das konkrete Vorgehen in Python	196
13.3.1	String-Konstanten und die Format Specification Mini-Language	196
13.3.2	String-Funktionen	197
13.3.3	String-Methoden	197
13.4	Umgang mit regulären Ausdrücken	198
13.4.1	Was sind allgemein reguläre Ausdrücke?	198
13.4.2	Wo setzt man reguläre Ausdrücke ein?	199
13.4.3	Details zu Pattern	200
13.4.4	Optionen für die Häufigkeit	204
13.4.5	Die Umsetzung von regulären Ausdrücken in Python – das Modul re.	205
13.4.6	Die Match-Objekte	206
13.4.7	Ein paar Beispiele mit regulären Ausdrücken	207
14	Datei-, Datenträger- und Datenbankzugriffe – Dauerhafte Daten	211
14.1	Was behandeln wir in diesem Kapitel?	211
14.2	Datenströme für die Ein- und Ausgabe	211
14.2.1	Das Öffnen und Schließen einer Datei	212
14.2.2	Schreiben in eine Datei	213
14.2.3	Auslesen aus einer Datei	213
14.2.4	Lese- und Schreibvorgänge absichern	215
14.3	Allgemeine Datei- und Verzeichnisoperationen	215
14.4	Objekte serialisieren und deserialisieren	217
14.4.1	Mit dump() den Objektzustand persistent machen	218
14.4.2	Mit load() den Objektzustand reproduzieren	218
14.5	Datenbankzugriffe	219
14.5.1	Was ist SQLite?	219
14.5.2	Zugriff auf SQLite in Python	219
14.5.3	Ein konkretes Datenbankbeispiel	221
15	Umgang mit Datum und Zeit – Terminsachen	225
15.1	Was behandeln wir in diesem Kapitel?	225
15.2	Allgemeines zum Umgang mit Datum und Zeit	225
15.3	Die Python-Module	226
15.4	Typische Beispiele für Operationen mit Zeit und Datum	226
15.4.1	Das aktuelle Systemdatum des Computers auslesen	227
15.4.2	Ein beliebiges Datumsobjekt erstellen	227

16 Grafische Oberflächen (GUI) mit Python – Das Modul tkinter als GUI-Framework	233
16.1 Was behandeln wir in diesem Kapitel?	233
16.2 Hintergrundinformationen zu modernen grafischen Oberflächen	233
16.3 Konkrete GUI-Konzepte in Python und das Modul tkinter	234
16.3.1 Ein Fenster vom Typ TK als Basis jeder GUI-Applikation	234
16.3.2 Der übliche OO-Ansatz	234
16.3.3 Die Geometry Manager	235
16.3.4 Wichtige GUI-Elemente	242
16.4 Die Ereignisbehandlung	242
16.4.1 Die konkrete Ereignisbehandlung in Python	243
16.5 Eine grafische Datenbankapplikation	245
Stichwortverzeichnis	249



Einleitung und Grundlagen – Bevor es richtig losgeht

1

1.1 Was behandeln wir in dem einleitenden Kapitel?

Bevor es mit Python richtig losgeht, sollen in diesem einleitenden Kapitel einige Dinge geklärt werden, die Ihnen die folgende Arbeit mit diesem Buch und der Programmiersprache im Allgemeinen erleichtern werden. Insbesondere sorgen wir an der Stelle dafür, dass Ihnen Python überhaupt zur Verfügung steht.

1.2 Das Ziel des Buchs

Dieses Buch ist zum Lernen von Python gedacht, entweder in Form des Selbststudiums oder als Begleitmaterial in entsprechenden Kursen. Zuerst einmal lernen Sie die elementaren Grundlagen, um überhaupt Programme mit Python erstellen, als auch pflegen zu können. Danach werden erweiterte Techniken wie umfassendere Anwendung der Syntax, objektorientierte Programmierung mit Python, komplexere Anwendungen mit sequenziellen Datenstrukturen, Umgang mit Modulen, Ausnahmebehandlung, Dateizugriffe, Datenbankzugriffe und die Erstellung grafischer Oberflächen behandelt. Dabei wird über sämtliche Themen hinweg Wert auf die grundsätzliche Anwendung der verschiedenen Techniken und einfache Beispiele gelegt und nicht auf Vollständigkeit aller möglichen Anweisungen, Befehle oder Parameter. Ebenso soll immer wieder darauf hingewiesen werden, dass man gewisse Freiheiten, die Ihnen Python bietet, nicht unbedingt ausnutzen sollte.

- ▶ **Tipp** Wir erstellen im Laufe des Buchs immer wieder praktische Beispiele. Die Quellcodes des Buchs finden Sie nach Kapiteln und darin erstellten Projekten sortiert auf den Webseiten des Verlags. Die Namen der jeweilig aktuellen Dateien beziehungsweise Projekte werden als Hinweise oder direkt im Text vor den jeweiligen Beispielen angegeben und bei Bedarf wiederholt. Ich empfehle allerdings, dass Sie die Beispiele unbedingt allesamt von Hand selbst erstellen. Das ist für das Verständnis und das Lernen eindeutig besser als ein reines Kopieren oder nur Anschauen.

1.3 Was sollten Sie bereits können?

Der Kurs ist als Einsteigerkurs konzipiert, in dem die Sprache Python von Grund auf erarbeitet wird. Dabei wird der Tatsache Rechnung getragen, dass Python mittlerweile oft als erste Programmiersprache überhaupt gelernt wird. Um nicht die einfachsten Programmiergrundlagen zu ausführlich erläutern zu müssen, wird aber zumindest ein Verständnis der Idee von Programmierung vorausgesetzt, obwohl Sie noch nicht zwingend programmieren müssen. Ebenso sollten Sie mit einem Texteditor umgehen können. Kenntnisse der wichtigsten Befehle Ihres Betriebssystems (Windows, Linux oder macOS) zum Umgang mit Dateien und Verzeichnissen in der Konsole sind hilfreich. Umsteiger aus anderen Sprachen sind aber auch explizit als Zielgruppe des Buchs einkalkuliert und entsprechende Hinweise auf andere Programmiersprachen werden immer wieder Bezüge herstellen – insbesondere zu Sprachen der Webprogrammierung (PHP, JavaScript, ...), Java und .NET-Sprachen.

1.4 Was ist Python?

Python ist eine universelle, höhere Programmiersprache, die üblicherweise interpretiert wird. Es gibt also in der Laufzeitumgebung von Python einen Interpreter samt notwendiger weiterer Ressourcen.

Hintergrundinformation

Nun ist der Begriff des Interpreters gefallen und die Konzepte und Fachausdrücke zur Übersetzung von Quellcode sollen nicht einfach vorausgesetzt werden. Wenn Sie Python-Programme erstellen, schreiben Sie den sogenannten **Quellcode** oder **Quelltext**, der der englischen Sprache angelehnt ist. Aber zur Ausführung muss dieser übersetzt werden, und zwar in ein Format, das ein Computer versteht und das ist **Maschinencode**. Dazu gibt es verschiedene Möglichkeiten.

- Einmal gibt es den **Interpreter**. Dieser sorgt für eine Übersetzung in Maschinencode während der Programmausführung. Interpreter übersetzen den Quellcode eines Programms zeilenweise.
- Die Alternative ist der **Compiler**, der den kompletten Quelltext vor der Laufzeit übersetzt.
- In den vergangenen Jahren hat sich auch eine **zweistufige Übersetzung** etabliert – die Übersetzung mit Zwischencode. Das wird in Java oder dem .NET-Framework gemacht. Denn ein Nachteil bei kompilierten Programmen ist, dass diese Programme maschinenabhängig sind

und somit nicht auf jeder Computerplattform, zum Beispiel Linux und Windows, ausgeführt werden können. In Java oder bei der .NET-Plattform wird Quellcode nicht zu einem ausführbaren Programm, sondern in einen Zwischencode, kompiliert (1. Schritt). Dieser Code ist für alle Plattformen gleich und kann mithilfe des entsprechenden plattformspezifischen Interpreters (2. Schritt) auf der jeweiligen Plattform ausgeführt werden.

1.4.1 Das Ziel von Python

Python ist den meisten gängigen Programmiersprachen verwandt,¹ wurde aber mit dem Ziel größter Einfachheit und Übersichtlichkeit entworfen. Zentrales Ziel bei der Entwicklung der Sprache ist die Förderung eines gut lesbaren, knappen Programmierstils. So wird beispielsweise der Code nicht durch geschweifte Klammern (wie in fast allen C-basierenden Sprachen), sondern durch zwingende Einrückungen strukturiert.² Zudem ist die gesamte Syntax reduziert und auf Übersichtlichkeit optimiert.

Wegen dieser klaren und überschaubaren Syntax gilt Python als einfach zu erlernen, zumal die Sprache mit relativ wenig Schlüsselwörtern auskommt. Es wird immer wieder zu hören sein, dass sich pythonbasierte Skripte deutlich knapper formulieren lassen als in anderen Sprachen.

1.4.2 Was umfasst Python?

Es gibt einmal die Sprache Python, die aus den üblichen Schlüsselworten, Operatoren, eingebauten Funktionalitäten etc. sowie einer eigenständigen Syntax besteht. Python besitzt zudem eine umfangreiche Standardbibliothek (API – Application Programming Interface, wörtlich „Anwendungsprogrammierschnittstelle“) und zahlreiche Pakete im Python Package Index, bei deren Entwicklung ebenfalls großer Wert auf Überschaubarkeit, aber auch eine leichte Erweiterbarkeit gelegt wurde.

Python-Programme lassen sich deshalb auch in anderen Sprachen als Module einbetten. Umgekehrt lassen sich mit Python Module³ und Plug-ins⁴ für andere Programme schreiben, die die entsprechende Unterstützung bieten. Dies ist zum Beispiel bei Blender, Cinema 4D, GIMP, Maya, OpenOffice beziehungsweise LibreOffice, PyMOL, SPSS, QGIS oder KiCad der Fall.

¹Die meisten aktuellen Programmiersprachen gehen von der Syntax auf C zurück.

²Was allerdings viele erfahrene Programmierer mit C-Background eher verstört, denn die saubere Notation von geschweiften Klammern ist sehr übersichtlich.

³Zusammenfassungen von Quellcodestrukturen. Modul steht allgemein für einen Teil eines größeren Systems.

⁴Plug-ins sind allgemeine kleine Programme oder Programmpakete, mit denen sich Software nach den eigenen Bedürfnissen anpassen und erweitern lässt. Sie integrieren sich dazu in das System, das sie erweitern oder anpassen sollen.

1.4.3 Die verschiedenen Python-Paradigma

Python unterstützt sowohl die objektorientierte, die aspektorientierte, die strukturierte als auch die funktionale Programmierung. Das bedeutet, Python zwingt den Programmierer nicht zu einem einzigen Programmierstil, sondern erlaubt, das für die jeweilige Aufgabe am besten geeignete Paradigma zu wählen. Objektorientierte und strukturierte Programmierung werden vollständig, funktionale und aspektorientierte Programmierung werden zumindest durch einzelne Elemente der Sprache unterstützt. Ein zentrales Feature ist in Python die dynamische Typisierung samt dynamischer Speicherbereinigung. Damit kann man Python auch als reine Skriptsprache nutzen.

1.5 Was benötigen Sie zum Arbeiten mit dem Buch?

Python steht auf verschiedenen Plattformen und in verschiedenen Versionen zur Verfügung. Wir schauen uns erst einmal an, mit welchen Voraussetzungen Sie an das Buch gehen können.

1.5.1 Hardware und Betriebssystem

Python gibt es für Windows und macOS sowie Linux. Damit sind die drei Betriebssysteme genannt, mit denen wohl die meisten Leser arbeiten werden. Aber es gibt Python auch für viele weitere, teils ziemlich alte beziehungsweise exotische Betriebssysteme (unter anderem AS/400 (OS/400), BeOS, MorphOS, MS-DOS, OS/2, RISC OS, Series 60, Solaris oder HP-UX), wobei Sie beachten sollten, dass für ältere Betriebssysteme die neuesten Python-Versionen oft nicht mehr zur Verfügung stehen. Ab der Version Python 3.5 wird etwa Windows XP nicht mehr unterstützt. Aber auch viele vorinstallierte Versionen von Python – wenn es denn solche gibt – sind nicht auf dem aktuellen Stand.

Letztendlich sind aber die Voraussetzungen für die Arbeit mit Python ziemlich niedrig. Sie benötigen für die Arbeit mit dem Buch und Python im Grunde nur einen Computer mit einem halbwegs modernen Betriebssystem. Als Basis für die Unterlagen wird explizit ein PC vorausgesetzt.

1.5.2 Die Python-Version

Die **Referenzbetriebssysteme** in diesem Buch werden Windows 10, macOS 10.3 sowie eine aktuelle Linux-Distribution sein.⁵ Überwiegend wird in den Unterlagen mit Windows 10 gearbeitet, aber die Ausführungen und Beispiele sind auf andere Systeme übertragbar oder nicht von der Plattform abhängig. Sollten Spezifika interessant sein, wird das hervorgehoben.

⁵ Konkret werden an einigen Stellen Mint Linux 18 und Ubuntu 17 verwendet.

1.5.2.1 Die Evolution von Python

- Python wurde Anfang der 1990er-Jahre von Guido van Rossum am Centrum Wiskunde & Informatica in Amsterdam als Nachfolger für die Sprache ABC entwickelt.
- Ursprüngliche Zielplattform war das verteilte Betriebssystem Amoeba.
- Die erste Vollversion erschien im Januar 1994.
- Python 2.0 erschien Oktober 2000.
- Python 3.0 (auch Python 3000) erschien im Dezember 2008 und die Serie 3 ist 2017 immer noch aktuell.

1.5.3 Python laden und installieren

Zum Erstellen von Python-Programmen braucht man im Grunde nur einen Klartexteditor. Aber um Python-Programme oder -Skripte auszuführen, braucht man mehr – eine Umgebung für Python. Deshalb muss für die weiteren Schritte mit dem Buch Python auf Ihrem Rechner installiert sein oder Sie sollten das zu Beginn machen. Nun müssen Sie beachten, dass es aktuell zwei „Schienen“ bei den Versionen von Python gibt. Zwar ist Python 3.0 wie erwähnt bereits im Dezember 2008 erschienen. Da Python 3.0 jedoch teilweise inkompatibel zu früheren Versionen ist, wird Python 2.7 derzeit parallel zu Python 3 weiter durch Updates unterstützt.

- ▶ Die Besonderheiten der 2er-Versionen von Python werden im Buch nicht mehr berücksichtigt.

Die **Referenzversion** von Python für dieses Buch ist die Version 3.6.x, wobei unsere Ausführungen im Grunde für alle Versionen 3.x gelten. Die Unterschiede sind für dieses Buch nicht wirklich relevant. Sie sollten auch beachten, dass es auch für unsere Referenzbetriebssysteme kleinere Abweichungen in der verfügbaren Version von Python geben kann. Oder genauer: Es kann sich zeitlich etwas unterscheiden, wann etwa eine Version für Windows, Linux oder macOS veröffentlicht wird.

1.5.3.1 Das zentrale Webportal und die Python Software Foundation (PSF)

Wenn Sie erstmals Kontakt zu Python aufnehmen wollen, gehen Sie am besten zuerst auf die Webseite <https://www.python.org/> (Abb. 1.1). Das ist das zentrale Webportal von Python. Hier finden Sie die wichtigsten Informationen zu Python als auch alle notwendigen Ressourcen.

Von da gelangen Sie über einen Link oben in der Navigation oder direkt über den Link <https://www.python.org/psf/> zur Python Software Foundation (PSF Abb. 1.2).

PSF ist eine Non-Profit-Organisation, die hinter dem Open-Source-Projekt der Programmiersprache steht. Mitglieder der PSF sind sowohl relevante Einzelpersonen aus dem Python-Umfeld, als auch Firmen wie Google, Microsoft, Redhat und Canonical, die als Sponsoren agieren. Die PSF ist Herausgeber und Rechteinhaber der Python-Software-Foundation-Lizenz (<https://docs.python.org/3/license.html>, Abb. 1.3) und besitzt die Markenrechte an Python. Die Python-Software-Foundation-Lizenz ist ähnlich der BSD-Lizenz und kompatibel mit der GNU General Public License.

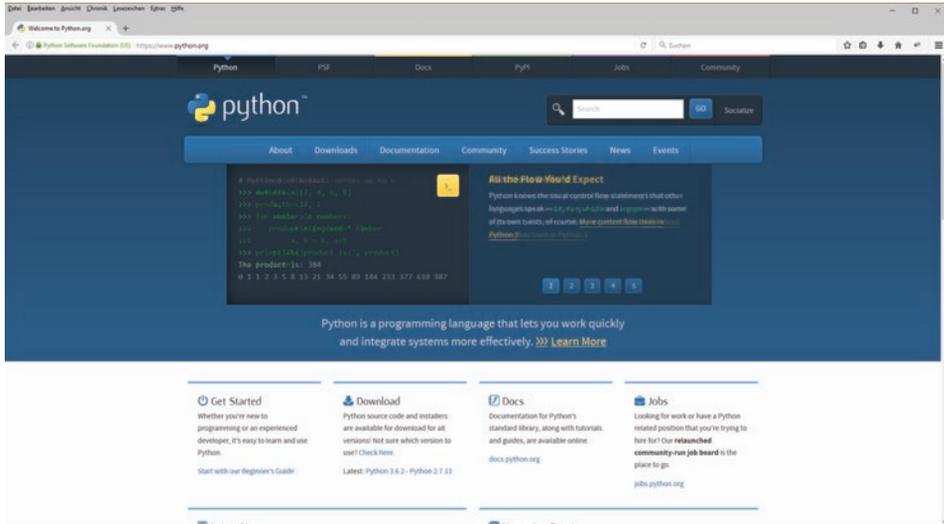


Abb. 1.1 Das „offizielle“ Webportal zu Python



Abb. 1.2 Die Webseite der PSF

1.5.3.2 Die Dokumentation

Im dem Python-Webportal finden Sie unter dem Menüpunkt DOCUMENTATION und dort Docs zahlreiche Informationen inklusive einer kompletten Dokumentation, FAQs und verschiedenen Tutorials (Abb. 1.4).

Von da aus kann man diese Materialien herunterladen, aber das ist im Grunde gar nicht notwendig. Zum einen ist man ja sowieso meist online und zudem ist die Dokumentation

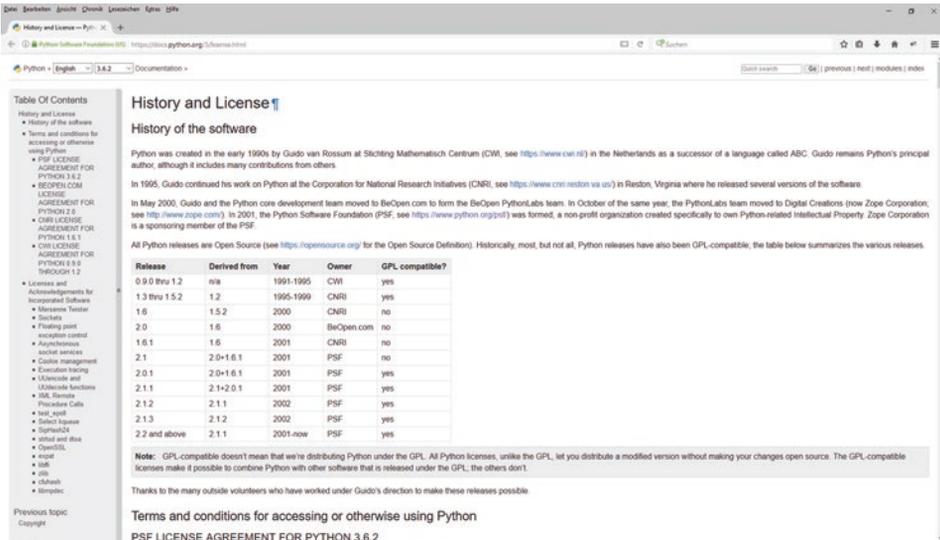


Abb. 1.3 Details zur Python-Lizenz

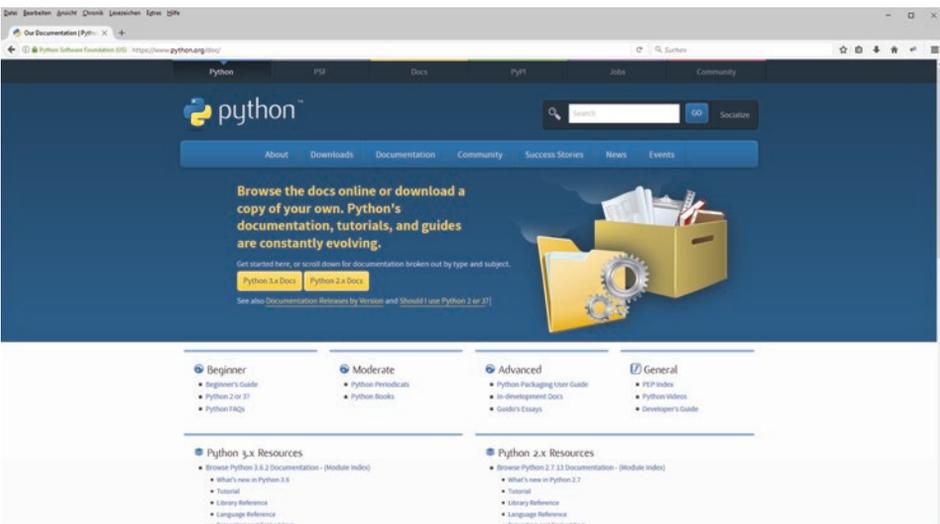


Abb. 1.4 Informationen, Quellen und Tutorials

bereits im Installationspaket von Python enthalten beziehungsweise wird mit installiert, wenn Sie das bei der Installation auswählen.

1.5.3.3 Der Download von Python

Bereits im Dokumentationsbereich steht Ihnen Python zum Download bereit. Aber es gibt auch auf der Einstiegsseite des Webportals einen eigenen Menüpunkt DOWNLOADS in

Kapitälchen. Darüber erhalten Sie die aktuellen, aber auch bei Bedarf frühere Versionen für verschiedene Betriebssysteme (Abb. 1.5).

- **Tipp** Laden Sie sich am besten immer die neueste Version von Python für Ihr Betriebssystem herunter.

1.5.3.4 Die konkrete Installation

Wir gehen nun die Installation für Windows, Linux und macOS einzeln durch und Sie werden sehen, dass diese in der Regel geradezu trivial ist, wenn es einen Installer gibt. Unter Linux und teils auch macOS kann es aber sein, dass Sie Python in einem Terminal installieren müssen, wenn Sie die aktuelle Version haben wollen.

1.5.3.4.1 Die Installation unter Windows

Unser wichtigstes Referenzsystem im Buch ist Windows und damit wollen wir beginnen. Unter Windows starten Sie einfach den Installer, den Sie auf Ihren PC geladen haben. Dieser startet wiederum mit einem Auswahldialog mit allen wichtigen Informationen und Vorgabeeinstellungen (Abb. 1.6).

Wenn Sie in dem Dialog bereits den Installationsbefehl geben (INSTALL NOW), läuft die Installation ohne weitere Interaktion mit Ihnen durch – abgesehen von eventuellen Rückfragen von Windows, zum Beispiel ob Sie dem Installationsprogramm wirklich erlauben wollen, Dinge auf dem PC zu verändern. Auf dem Rechner hat der Python-Installer danach alle Dateien im voreingestellten Verzeichnis installiert, die man für die Ausführung von Python-Programmen benötigt.

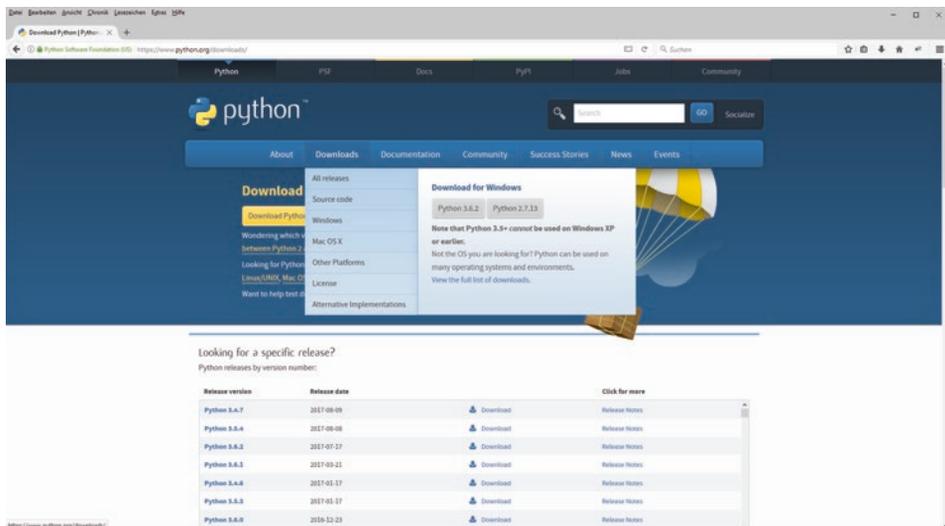


Abb. 1.5 Python laden



Abb. 1.6 Start des Setupassistenten

Nun gibt es in dem Dialog zwei optionale Einstellungsmöglichkeiten, die Sie beide auswählen sollten.

1. Die erste Option stellt allen Benutzern auf dem Rechner den sogenannten Launcher für Python zur Verfügung. Das bedeutet, dass diese Anwender dann Python-Dateien direkt ausführen können. Oder anders ausgedrückt: Die Python-Dateierweiterung wird mit dem Python-Interpreter verknüpft.
 2. Besonders zu empfehlen ist das Hinzufügen von Python zur *Path*-Angabe – dem Suchpfad von Windows. Dazu klicken Sie in dem Dialog die entsprechende Option an – bei der im Buch verwendeten Version von Python nennt die sich `ADD PYTHON 3.6 TO PATH`. Wenn Sie die entsprechende Option ausgewählt haben, wird das Installationsverzeichnis in den Suchpfad von Windows aufgenommen und Sie können Python in der Konsole von jedem Verzeichnis aus aufrufen. Das ist zwar nicht zwingend notwendig, aber sehr bequem.
- **Tipp** Sie können die Installationsverzeichnisse unter Windows selbstverständlich nachträglich dem Suchpfad hinzufügen. Das macht man in den Einstellungen zum System unter `ERWEITERTE UMGEBUNGS-VARIABLEN`. Aber wenn das vom Setup-Assistent schon geleistet wird, ist das natürlich angenehm.

Nach der Installation findet Windows insbesondere zwei Programme, die von zentraler Bedeutung sind für Python:

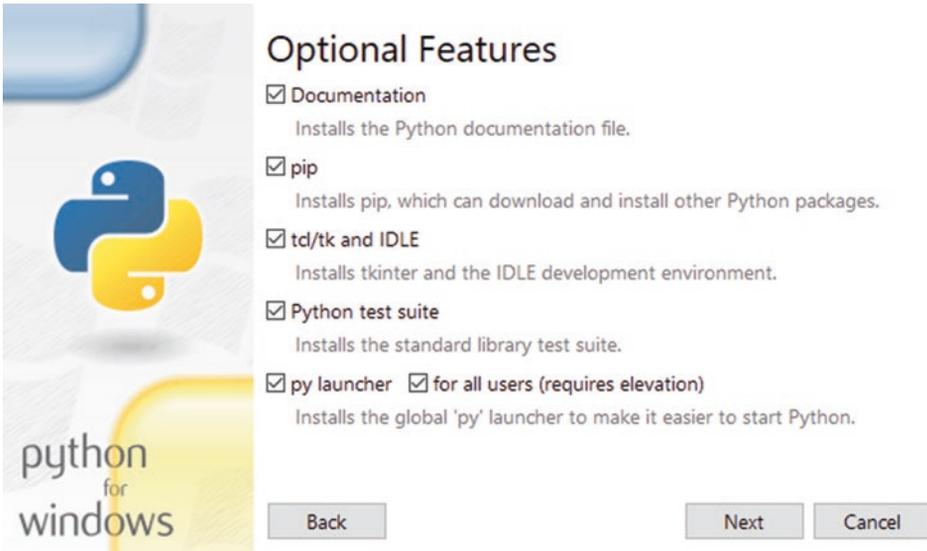


Abb. 1.7 Optionale Features

- Die **Python-Kommandozeile** – auch Python-Interpreter⁶ oder Python-Shell genannt,
- die Python-GUI, die man mit **IDLE** (Python's Integrated Development Environment) abkürzt.

Auf die Details dazu gehen wir später ein.

Wenn Sie wollen, können Sie vom Einstiegsdialog aus die Arbeit des Assistenten auch individuell anpassen.

Einmal können Sie optionale Features festlegen (Abb. 1.7). Diese Einstellungen können aber meist in der Voreinstellung bleiben.

Aber sehr interessant sind die erweiterten Optionen im darauffolgenden Anpassungsdialog (Abb. 1.8). Denn insbesondere die Änderung des Installationsorts von Python kann wichtig sein. Ich verwende etwa bei meinem Windows-System aktuell eine Workstation mit einer recht kleinen SSD, aber einer sehr großen HDD. Die Daten und nicht wirklich performancehungrige Programme lagere ich deshalb grundsätzlich auf die größere HDD aus. Und Python benötigt in der Regel definitiv nicht wirklich viel Performance und liegt bei mir auf dem Laufwerk F (der HDD).

Wenn die Installation beendet ist, erhalten Sie eine Erfolgsmeldung (Abb. 1.9).

Wenn ein Fehler passiert ist (was selbstverständlich weder zu hoffen, noch zu erwarten ist), werden Sie natürlich auch über die Art der Probleme informiert.

⁶Was eigentlich etwas ungenau ist, denn der Interpreter an sich ist keine Konsole in dem Sinn. Aber man benutzt den Begriff dennoch im Python-Umfeld.

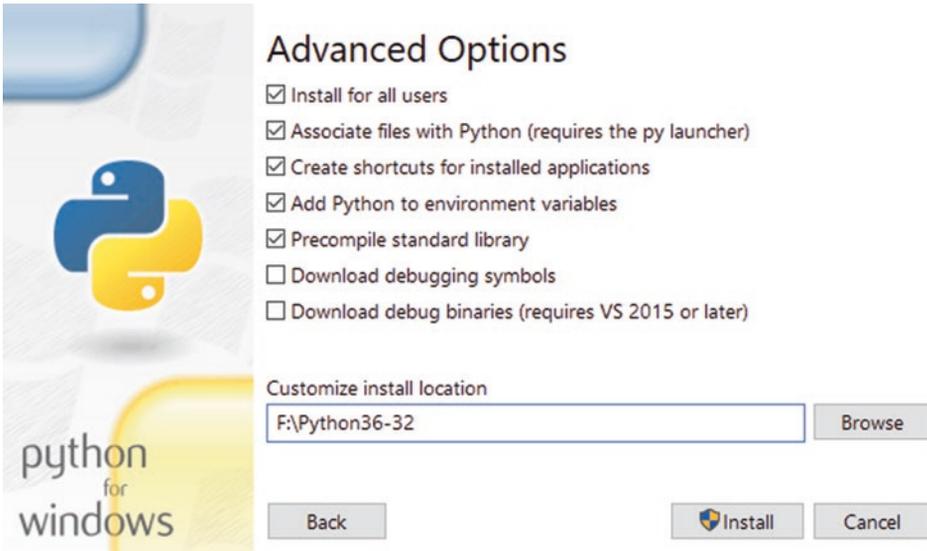


Abb. 1.8 Erweiterte Optionen

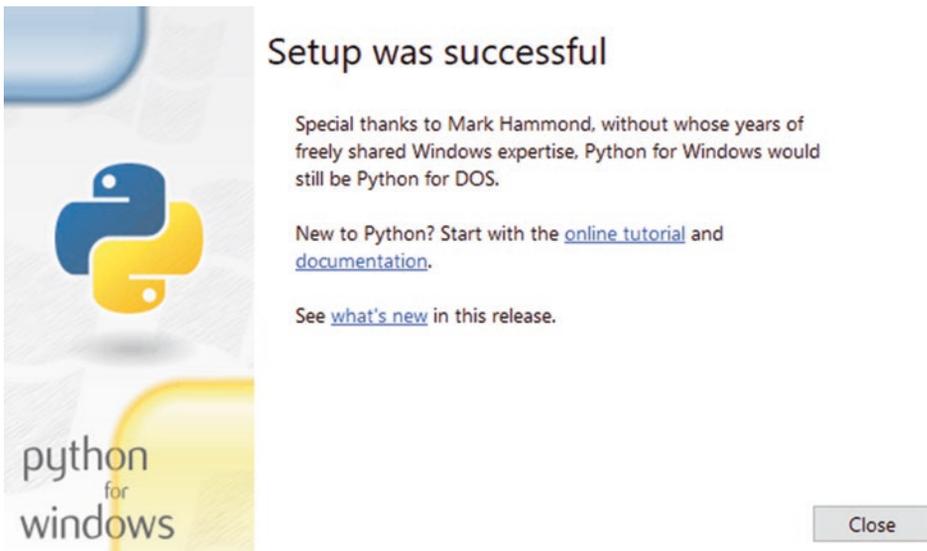


Abb. 1.9 Python wurde installiert

1.5.3.4.2 Installieren auf einem Linux-System

Python gibt es wie gesagt auch für Linux. Dabei muss man einmal einschränken, dass es verschiedene Formate für die unterschiedlichen Linux-Distributionen gibt. Das macht die Sache etwas unübersichtlich. Aber dafür haben Sie auf der anderen Seite die Flexibilität und können Python sogar individuell angepasst aus Quellcodes für Ihr System erstellen.

Wobei das im Prinzip meist gar nicht notwendig ist. Denn viele Linux-Distributionen installieren Python bereits automatisch mit. Oder sie stellen Python über die distributions-eigene Anwendungsverwaltung zur Verfügung, womit die Installation genauso trivial wie unter Windows abläuft (Abb. 1.10).

Allerdings muss man beachten, dass in der Regel nicht die aktuelle Version von Python in den Installationspaketen beziehungsweise der standardmäßig installierten Version bereitsteht. Sogar aktuelle Linux-Versionen wie Mint 18 (Abb. 1.11) oder Ubuntu 17 (Abb. 1.12) stellen da noch Python in der Version 2.7.x bereit und die Besonderheiten der 2er-Versionen werden wir wie gesagt explizit nicht mehr beachten. Wie erwähnt, liegt Python zum Zeitpunkt der Bucherstellung in der Version 3.6.x vor und diese sollten Sie auch unter Linux zur Verfügung haben.

Falls Sie also die neueste Version oder zumindest eine Version der 3er-Schiene von Python nicht über die Anwendungsverwaltung Ihrer Linux-Distribution erhalten, können Sie eine individuelle Installationsdatei oder auch die Quellcodes laden und dann auf Ihrem Rechner konfigurieren, übersetzen und installieren (Abb. 1.13).



Abb. 1.10 Linux-Distributionen bringen Python eigentlich immer mit

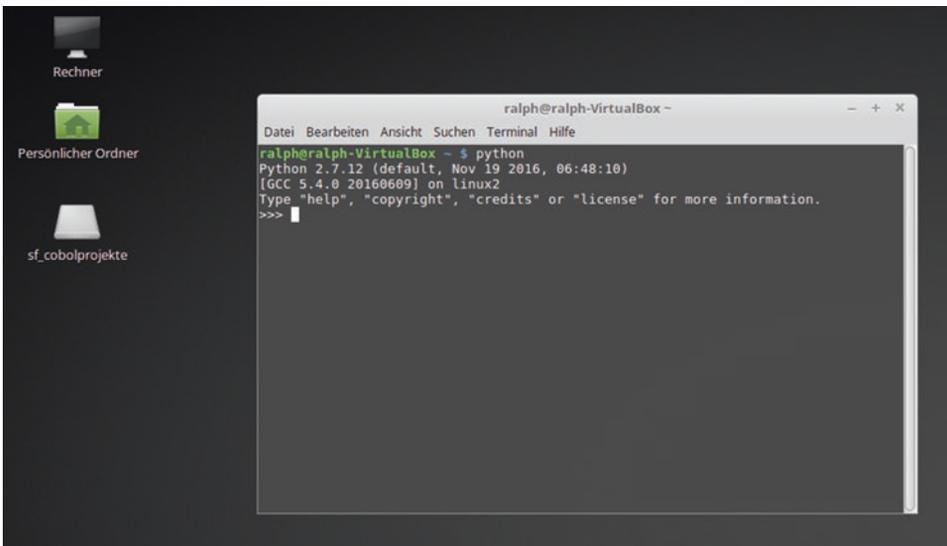
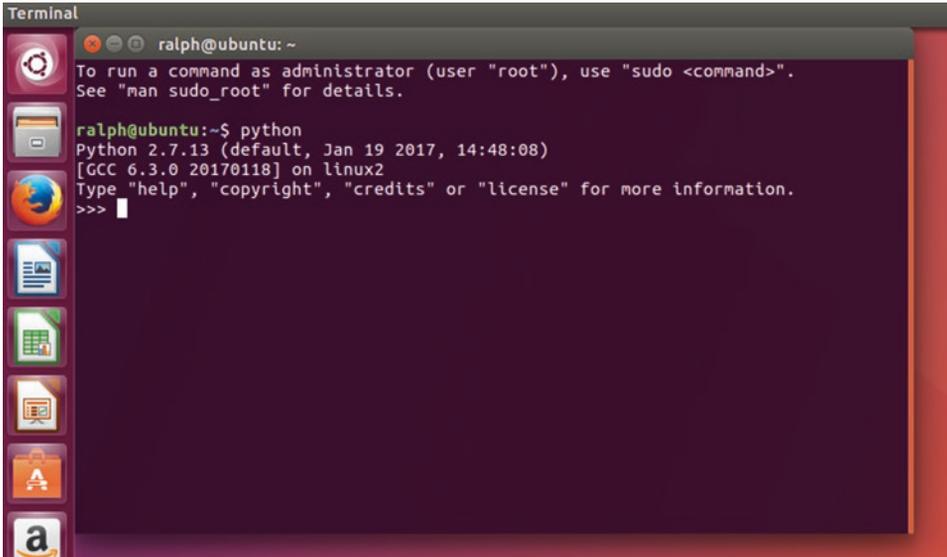


Abb. 1.11 Unter Mint Linux 18 ist noch die Version 2.7.12 vorinstalliert



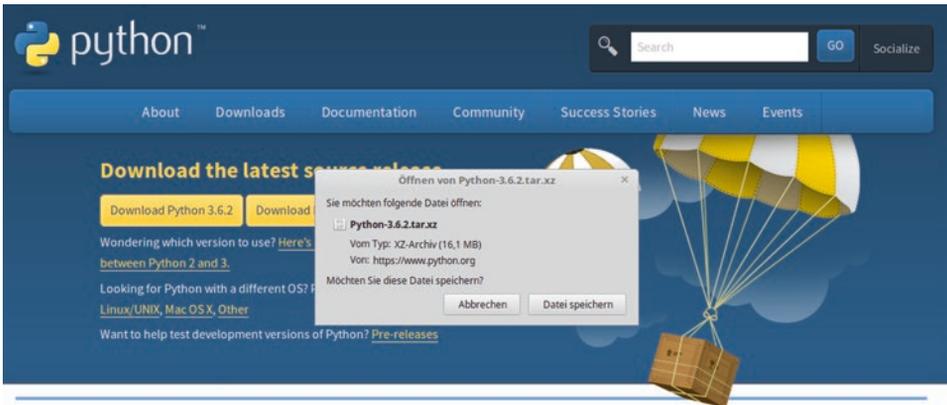
```

Terminal
ralph@ubuntu: ~
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ralph@ubuntu:~$ python
Python 2.7.13 (default, Jan 19 2017, 14:48:08)
[GCC 6.3.0 20170118] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>

```

Abb. 1.12 Auch Ubuntu ist nur marginal aktueller



Looking for a specific release?

Python releases by version number:

Abb. 1.13 Python für Linux laden

Wie die dann aber konkret installiert wird, kann sich unterscheiden und es muss für Details zu einer bestimmten Distribution auf die Dokumentation verwiesen werden – gerade im Fall von Problemen. Aber normalerweise erhalten Sie auf jeden Fall die Quelltextdateien von Python, die Sie auf die übliche Art unter Linux konfigurieren und installieren können. Die Ressourcen liegen dabei als Archiv vor, das Sie erst einmal extrahieren müssen (Abb. 1.14).

Danach öffnen Sie ein Terminal und gehen im allgemeinen Fall wie folgt vor (Abb. 1.15):

1. Wechseln Sie in das Verzeichnis, das bei der Extraktion der Python-Ressourcen angelegt wurde, etwa so: