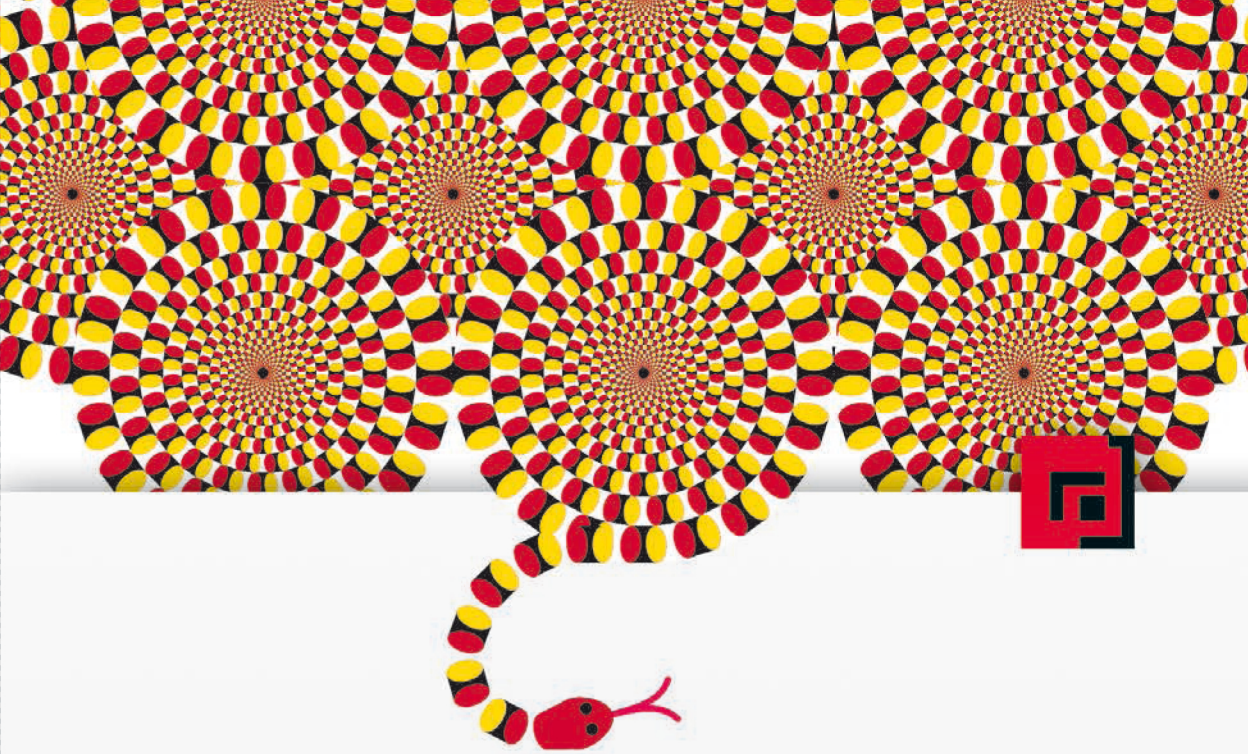


Jason R. Briggs

Python kinderleicht!

Einfach programmieren lernen – nicht nur für Kids

US-Bestseller

dpunkt.verlag

Jason Briggs

Python kinderleicht!

Einfach programmieren lernen – nicht nur für Kids

2., korrigierte und aktualisierte Auflage

Übersetzung aus dem Amerikanischen
von Volker Haxsen

dpunkt.verlag

Lektorat: Dr. Michael Barabas
Übersetzung und Aktualisierungen: Volker Haxsen, Heidelberg
Copy-Editing: Friederike Daenecke, Zülpich
Herstellung: Nadine Thiele
Illustrationen: Miran Lipovača
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

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.

ISBN:
Buch 978-3-86490-344-1
PDF 978-3-86491-904-6
ePub 978-3-86491-905-3
mobi 978-3-86491-906-0

2., korrigierte und aktualisierte Auflage 2016

Titel der Originalausgabe: Python For Kids – A Playful Introduction To Programming
No Starch Press, Inc. · 38 Ringold Street, San Francisco, CA 94103 · http://www.nostarch.com/
ISBN 978-1-59327-407-8

5 4 3 2 1 0

Vorwort

Über den Autor

Jason R. Briggs ist seit dem Alter von acht Jahren Programmierer und hat als erste Programmiersprache BASIC auf einem Radio Shack TRS-80 erlernt. Er hat als Entwickler und Systemarchitekt professionell Software programmiert und als Autor für das *Java Developers's Journal* gearbeitet. Seine Artikel sind in *JavaWorld*, *ONJava* und *ONLamp* erschienen. *Python kinderleicht* ist sein erstes Buch.

Du kannst mit Jason über seine Homepage *http://jasonrbriggs.com/* oder per E-Mail *mail@jasonrbriggs.com* Kontakt aufnehmen.

Über die Fachkorrektoren

Der 15-jährige **Josh Pollock** ist frischgebackener Absolvent der *The Nueva School* und jetzt neu auf der *Lick-Wilmerding High School* in San Francisco. Er fing im Alter von neun Jahren mit dem Programmieren in Scratch an, begann in der sechsten Klasse mit TI-BASIC, ging dann in der siebten Klasse zu Java und Python über und machte in der achten Klasse mit UnityScript weiter. Neben dem Programmieren spielt er Trompete, entwickelt Computerspiele und unterrichtet Leute in MINT-Fächern.

Maria Fernandez hat einen Masterabschluss in angewandter Linguistik und interessiert sich schon seit über 20 Jahren für Computer und Technik. Sie hat jungen Flüchtlingsfrauen im *Global-Village*-Projekt in Georgia (USA) Englisch beige-

bracht, lebt zurzeit in Nord-Kalifornien und arbeitet für den ETS (Educational Testing Service).

Danksagungen

So ungefähr muss es sein, wenn man beim Empfang einer Ehrung die Bühne betritt und dann feststellt, dass man die Liste der Personen zu Hause hat liegen lassen, die man bei seiner Danksagung berücksichtigen will: Man vergisst garantiert jemanden, und die Musik setzt ganz schnell ein, um einen von der Bühne herunterzukomplimentieren.

Deswegen kommt jetzt eine (zweifelsohne) unvollständige Liste von Leuten, denen ich zu tiefem Dank verpflichtet bin, da sie mir geholfen haben, das Buch so gut werden zu lassen, wie es jetzt ist.

Ich möchte dem Team von No Starch danken, vor allem Bill Pollock, für seine bei der Bearbeitung immer wieder gestellte Frage, was denn ein Kind von alldem halten würde.

Wenn man schon sehr lange programmiert, vergisst man nur allzu leicht, wie schwer diese Dinge für Anfänger sind, und Bill war eine wertvolle Hilfe, weil er mich auf diese oft übersehenen und überkomplizierten Passagen aufmerksam machte. Mein Dank gilt auch Serena Yang, der exzellenten Produktionsmanagerin. Ich hoffe, dass sie sich nicht allzu sehr die Haare gerauft hat, als sie die richtige Farbgebung des Codes auf über 300 Seiten überprüfen musste.

Ein großes Dankeschön geht an Miran Lipovaca für ihre überaus gelungenen Illustrationen. Sie sind viel mehr als nur gelungen. Nein ehrlich! Wenn ich das gemacht hätte, könnte man von Glück sagen, wenn man ab und zu eine hingeschmierte Figur erkennen könnte. Ist es ein Bär? Ist es ein Hund? Nein, warte … soll das ein Baum sein?

Vielen Dank den Korrektoren! Ich muss mich dafür entschuldigen, dass nicht alle Vorschläge am Ende berücksichtigt wurden. Wahrscheinlich hattet Ihr recht, und ich kann nur eine schlechte Charaktereigenschaft von mir dafür verantwortlich machen, falls noch Fehler enthalten sind. Besonderer Dank geht an Josh für einige wirklich tolle Vorschläge und Ideen. Mein Bedauern gilt Maria, weil sie sich mit zum Teil uneinheitlich formatiertem Code herumschlagen musste.

Ich danke meiner Frau und meiner Tochter dafür, dass sie sich mit einem Mann und Vater abfinden mussten, der sich noch mehr als sonst hinter dem Computerbildschirm versteckt hat.

Meiner Mutter danke ich für all die unermüdliche Aufmunterung über all die Jahre.

Und zu guter Letzt danke ich meinem Vater dafür, dass er sich damals in den 1970er-Jahren einen Computer gekauft hat und es ertragen hat, dass ich diesen genauso oft nutzen wollte wie er. Nichts von alledem wäre ohne ihn möglich gewesen.

Inhaltsübersicht

Inhaltsverzeichnis

1 Einleitung

Warum soll man das Programmieren erlernen?

Programmieren fördert die Kreativität, das logische Denken und die Fähigkeit, Probleme zu lösen. Programmierer und Programmiererinnen haben die Möglichkeit, etwas aus dem Nichts zu erschaffen. Mithilfe der Logik bringen sie Programmstrukturen in eine Form, sodass ein Computer damit funktioniert. Und wenn die Dinge nicht ganz so gut funktionieren wie erwartet, können sie durch die Fähigkeit zur Problemlösung herausfinden, was schiefgelaufen ist. Programmieren macht Spaß, ist manchmal schwierig (gelegentlich frustrierend), und die Fähigkeiten, die man dabei erwirbt, können sowohl in der Schule als auch bei der Arbeit nützlich sein – selbst wenn Dein Berufsleben später nichts mit Computern zu tun haben sollte.

Außerdem ist das Programmieren ein prima Zeitvertreib bei miesem Wetter.

1.1 Warum Python?

Python ist eine leicht zu erlernende Programmiersprache, die für den Programmieranfänger einige nützliche Eigenschaften hat. Der Code ist im Vergleich zu anderen Programmiersprachen recht einfach zu lesen, und es gibt eine interaktive Shell, in die man seine Programme eingeben und sehen kann, wie sie laufen. Zusätzlich zu seiner einfachen Programmstruktur und seiner interaktiven Shell hat Python einige Merkmale, die den Lernvorgang sehr bereichern und mit denen Du einfache Animationen zum Erstellen Deiner eigenen Spiele zusammenbauen

kannst. Eines davon ist das Modul `turtle`, das von Turtle Graphics inspiriert wurde (das in den 1960er-Jahren von der Programmiersprache Logo verwendet wurde) und für Lernzwecke geschaffen wurde. Ein weiteres Modul ist `tkinter`, mit dem man auf das Tk GUI Toolkit zugreifen kann, um damit ziemlich einfach ein bisschen anspruchsvollere Grafiken und Animationen zu erstellen.

1.2 Wie man das Programmieren lernt

Wie bei allem, was man zum ersten Mal probiert, ist es am besten, mit den Grundlagen anzufangen. Beginne daher mit den ersten Kapiteln, und blättere nicht voller Ungeduld zu den Kapiteln weiter hinten. Niemand kann beim ersten Mal, wenn er ein Musikinstrument in die Hand nimmt, im Sinfonieorchester mitspielen. Flugschüler fliegen auch nicht, bevor sie die grundlegenden Steuerelemente verstanden haben, und Turner kriegen (normalerweise) beim ersten Versuch keinen Salto rückwärts hin. Wenn Du zu Anfang zu ungeduldig bist, haben die grundlegenden Prinzipien keine Zeit, sich richtig in Deinem Kopf festzusetzen. Dir wird dann der Inhalt der Kapitel weiter hinten viel komplizierter vorkommen, als er in Wirklichkeit ist.

Während Du dieses Buch durchliest, solltest Du jedes Beispiel selbst ausprobieren, um zu sehen, wie es funktioniert. Am Ende der meisten Kapitel gibt es auch Programmier-Puzzles, die Du lösen kannst. Sie werden Deine Programmierfähigkeiten fördern. Denke immer daran: Je besser Du die Grundlagen verstanden hast, desto leichter werden Dir die komplizierteren Konzepte später vorkommen.

Wenn Dich etwas frustriert oder Dir zu schwierig vorkommt, hier ein paar Ratschläge, die ich sehr hilfreich finde:

- Teile das Problem in kleinere Teile auf. Versuche zu verstehen, was ein kleiner Teil des Codes macht, oder denke nur an einen kleinen Teil einer komplexen Stelle. (Konzentriere Dich lieber auf einen kleinen Teil des Codes, statt alles auf einmal verstehen zu wollen.)
- Wenn das alles nichts hilft, ist es manchmal am besten, wenn man es für eine Weile einfach liegen lässt. Schlafe drüber, und mache an einem anderen Tag weiter. Auf diese Weise lösen sich viele Probleme von allein – besonders Programmierprobleme.

1.3 Wer dieses Buch lesen sollte

Dieses Buch ist für jeden geschrieben, der sich für das Programmieren interessiert, ganz egal, ob man nun Kind oder Erwachsener ist, wenn man zum ersten Mal programmiert. Wenn man lernen will, wie man seine eigene Software schreibt, anstatt nur von anderen entwickelte Programme zu nutzen, ist *Python kinderleicht* ein toller Einstieg.

In den folgenden Kapiteln erfährst Du, wie man Python installiert, die Python-Shell startet, einfache Berechnungen anstellt, Text auf den Bildschirm bekommt und Listen erstellt. Du lernst, wie man einfache Fallunterscheidungen mit `if`-Anweisungen und `for`-Schleifen durchführt. (Und natürlich erfährst Du, was `if`-Anweisungen und `for`-Schleifen eigentlich sind!) Du erfährst, wie man Code mit Funktionen wiederverwendet. Du lernst die Grundlagen von Klassen und Objekten kennen und bekommst Beschreibungen der vielen in Python eingebauten Funktionen und Module.

Es gibt Kapitel über einfache und fortgeschrittene Turtle-Grafiken und über die Benutzung des Moduls `tkinter`, um auf dem Computerbildschirm zu zeichnen. Am Ende vieler Kapitel gibt es Programmier-Puzzles mit unterschiedlichen Schwierigkeitsgraden, die dabei helfen, das gerade Gelernte zu verfestigen. Sie bieten Dir auch die Möglichkeit, selbst kleine Programme zu schreiben.

Wenn Du Dir die Grundlagen des Programmierens angeeignet hast, wirst Du lernen, wie Du Deine eigenen Spiele schreiben kannst. Du wirst zwei grafische Spiele entwickeln und etwas über Kollisionsdetektion, Events und diverse Animationstechniken erfahren.

Die meisten Beispiele in diesem Buch benutzen die IDLE-Shell (*Integrated DeveLopment Environment*; integrierte Entwicklungsumgebung) von Python. IDLE bietet Syntax-Markierung, eine Kopieren- und Einfügen-Funktionalität (so, wie Du es von anderen Anwendungen kennst) und ein Editor-Fenster, in dem Du Deinen Code für den späteren Gebrauch speichern kannst. IDLE ist daher eine Entwicklungsumgebung zum Experimentieren und hat auch ein bisschen was von einem Text-Editor. Die Beispiele funktionieren genauso gut in der Standard-Konsole und in einem üblichen Text-Editor, aber die Syntax-Markierung und die benutzerfreundlichere Umgebung von IDLE helfen Dir, den Code schneller zu verstehen. Deshalb wird im ersten Kapitel erklärt, wie man IDLE einrichtet.

1.4 Was in diesem Buch steht

Hier ist ein kurzer Überblick, was Dich in den einzelnen Kapiteln erwartet:

- **Kapitel 2** ist eine Einführung in das Programmieren. Außerdem findest Du Anleitungen zur ersten Installation von Python.
- **Kapitel 3** führt einfache Berechnungen und Variablen ein.
- **Kapitel 4** erklärt einige der grundlegenden Python-Elemente, wie etwa Strings, Listen und Tupel.
- **Kapitel 5** bietet Dir einen Vorgeschmack auf das Modul `turtle`. Wir springen dabei von den Grundlagen des Programmierens zum Bewegen einer Schildkröte (engl. *turtle*, die aber hier die Form eines Pfeils hat) über den Bildschirm.
- **Kapitel 6** behandelt die Varianten der Bedingungen und `if`-Anweisungen, und **Kapitel 7** macht bei den `for`- und `while`-Schleifen weiter.

- In **Kapitel 8** beginnen wir mit der Benutzung und Erstellung von Funktionen, und in **Kapitel 9** geht es um Klassen und Objekte. Wir decken in diesen beiden Kapiteln so viel von den grundsätzlichen Prinzipien der Programmiertechniken ab, dass wir in den weiteren Kapiteln zur Spiele-Entwicklung übergehen können. Von dort an wird es ein bisschen komplizierter.
- **Kapitel 10** stellt die meisten der eingebauten Funktionen von Python vor, und **Kapitel 11** macht mit ein paar Modulen (die im Prinzip Behälter voller nützlicher Funktionalität sind) weiter, die automatisch mit Python installiert wurden.
- **Kapitel 12** kehrt zum `turtle`-Modul zurück, da Du jetzt lernst, mit komplexeren Formen umzugehen. **Kapitel 13** geht zum Modul `tkinter` über – und damit zu fortgeschritteneren grafischen Kreationen.
- In den **Kapiteln 14** und **15** programmieren wir unser erstes Spiel, »Bounce!«, das auf dem Erlernten aus den vorigen Kapiteln aufbaut.
- In den **Kapiteln 16** bis **19** programmieren wir unser zweites Spiel: »Mr. Stick – Man rennt zum Ausgang.« In den Spieleentwicklungs-Kapiteln können die Dinge aus dem Ruder laufen. Wenn nichts mehr geht, lädst Du den Code von der Website zu diesem Buch (*www.dpunkt.de/python)* herunter und vergleichst Deinen Code mit den funktionierenden Beispielen von dort.
- Im **Nachwort** fassen wir das Gelernte mit einem Blick auf PyGame und andere beliebte Programmiersprachen zusammen.
- Zum Schluss sind im **Anhang** noch einmal alle Python-Schlüsselwörter genau erklärt, und im **Glossar** findest Du alle Definitionen der Programmierbegriffe, die in diesem Buch verwendet werden.

1.5 Die Website zum Buch

Wenn Du meinst, dass Du während des Lesens Hilfe brauchst, kannst Du die Website *www.dpunkt.de/python* aufsuchen, wo Du Downloads für alle Beispiele in diesem Buch und noch mehr Programmier-Puzzles findest. Du findest dort auch die Lösungen für alle Programmier-Puzzles in diesem Buch, falls Du nicht mehr weiter weißt oder Deine Programme überprüfen möchtest.

1.6 Viel Vergnügen!

Vergiss beim Durcharbeiten dieses Buches nie, dass Programmieren Spaß machen kann. Sieh es nicht als Arbeit an: Das Programmieren ist eine Möglichkeit, lustige Spiele oder Anwendungen zu erzeugen, die Du mit Deinen Freunden oder anderen teilen kannst.

Programmieren zu lernen ist ein tolles Training fürs Gehirn, und die Ergebnisse können sehr bereichernd sein. Aber vor allem gilt: Egal was Du tust, hab Spaß dabei!

Teil I

Programmieren lernen

2
Nicht alle Schlangen schlängeln sich

Ein Computerprogramm ist eine Gruppe von Anweisungen, die einen Computer dazu bringen, irgendetwas Bestimmtes zu machen. Es geht uns hier nicht um die physischen Bestandteile eines Computers, also die Drähte, Mikrochips, Karten, die Festplatte usw., sondern um die verborgenen Dinge, die auf dieser Hardware laufen. Ein Computerprogramm, das ich meist einfach nur *Programm* nenne, ist diese Gruppe von Befehlen, die der dummen Hardware sagt, was sie zu tun hat. Die *Software* ist eine Sammlung von Computerprogrammen.

Ohne Computerprogramme würde fast jedes Gerät, das wir täglich nutzen, entweder gar nicht funktionieren oder wäre weit weniger nützlich. In der einen oder anderen Form steuern Computerprogramme nicht nur Deinen Computer, sondern auch Videospiele, Mobiltelefone und Navigationsgeräte in Autos. Auch bei weniger offensichtlichen Dingen wie Flachbild-Fernsehern und deren Fernbedienungen sowie modernen Radios, DVD-Playern, Herden und einigen Kühlschränken übernimmt eine Software die Steuerung. Sogar Automotoren, Ampeln, die Straßenbeleuchtung, Zugsignale, elektronische Anzeigetafeln und Aufzüge werden von Programmen geregelt.

Programme sind ein bisschen wie Gedanken. Wenn Du keine Gedanken hättest, würdest Du wahrscheinlich nur auf dem Boden sitzen und auf Dein T-Shirt sabbern. Dein Gedanke »Stehe auf!« ist eine Anweisung, die Deinem Körper sagt, dass er aufstehen soll. Genauso sagen Computerprogramme dem Computer, was er zu tun hat.

Wenn Du weißt, wie man Computerprogramme schreibt, kannst Du allerlei nützliche Dinge anstellen. Sicherlich kannst Du dann nicht direkt Programme schreiben, die Autos, Ampeln oder Deinen Kühlschrank steuern, aber Du kannst damit Webseiten erzeugen, Deine eigenen Spiele programmieren oder Dir ein Programm schreiben, das Dir bei den Hausaufgaben hilft.

2.1 Ein paar Bemerkungen zum Thema Sprache

Wie wir Menschen auch benutzen Computer verschiedene Sprachen, um zu kommunizieren – nämlich Programmiersprachen. Eine *Programmiersprache* ist einfach eine bestimmte Art, mit dem Computer zu reden – eine Art, Anweisungen zu benutzen, die sowohl der Mensch als auch der Computer verstehen.

Es gibt Programmiersprachen, die nach Leuten benannt wurden (z.B. Ada und Pascal), solche, die Abkürzungen darstellen (z.B. BASIC und FORTRAN) und sogar solche, die wie Python nach Fernsehsendungen benannt wurden.

Ja, die Programmiersprache Python wurde nach der Sendung *Monty Python's Flying Circus* benannt und nicht nach der Python-Schlange.

Achtung!

Monty Python's Flying Circus war eine britische Comedy-Sendung, die in den 1970er-Jahren das erste Mal ausgestrahlt wurde. Sie ist bis heute bei einigen sehr beliebt. Die Show enthielt Sketche wie »The Ministry of Silly Walks«, »The Fish-Slapping Dance« und »The Cheese Shop« (in dem überhaupt kein Käse verkauft wurde). Mittlerweile sind Monty Python wohl durch ihre Spielfilme »Das Leben des Brian« und »Die Ritter der Kokosnuss« bekannter.

Eine ganze Reihe von Eigenschaften der Programmiersprache Python machen sie für Anfänger besonders geeignet. Die wichtigste Eigenschaft ist, dass Du mit Python ziemlich schnell einfache, aber wirkungsvolle Programme schreiben kannst. Python verwendet nicht viele dieser komplizierten Zeichen, wie geschweifte Klammern ({ }), Doppelkreuze (#) oder Dollarzeichen($), die andere Programmiersprachen viel schwerer zu lesen machen und daher auf Anfänger abschreckend wirken.

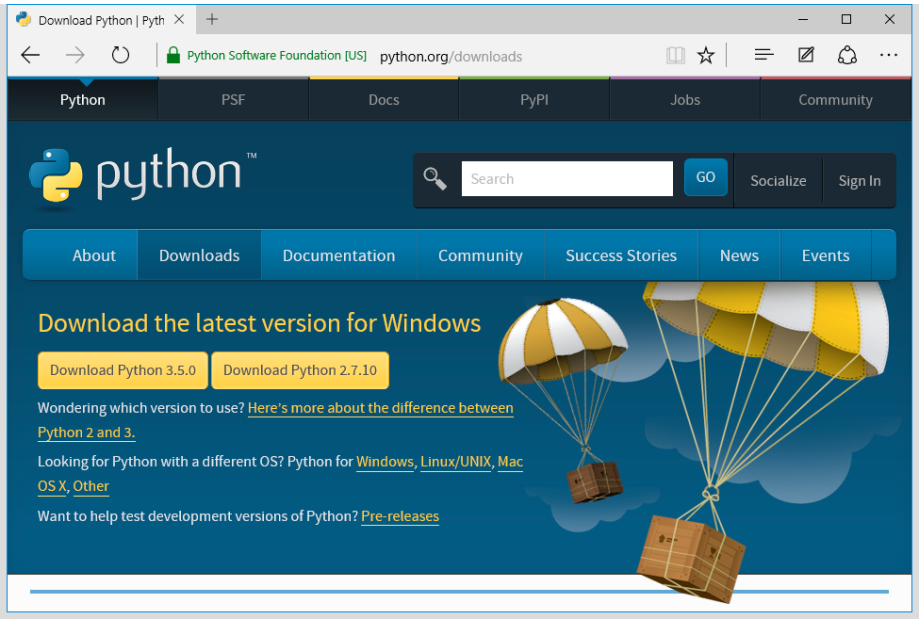
2.2 Python installieren

Die Installation von Python ist sehr unkompliziert. Wir gehen hier die Schritte der Installation in Windows, MacOSX und Ubuntu durch. Beim Installieren von Python legst Du Dir auch eine Verknüpfung zum Programm IDLE an. Das ist die integrierte Entwicklungsumgebung, in der Du später Deine Programme schreiben kannst.

Falls Python schon auf Deinem Computer installiert ist, kannst Du zu Abschnitt 2.3 weiterblättern.

Python unter Windows installieren

Um Python für Microsoft Windows zu installieren, gehst Du mit Deinem Browser auf *http://www.python.org* und lädst Dir den aktuellen Windows-Installer für Python 3 herunter.



Achtung!

Welche Python-Version genau Du herunterlädst, ist nicht entscheidend, solange vorne eine 3 steht.

Nachdem Du den Windows-Installer heruntergeladen hast, machst Du einen Doppelklick auf sein Icon und folgst dann den Anweisungen, um Python an seinem voreingestellten Speicherort wie folgt zu installieren:

1. Wähle **Install for all Users**, und klicke unten auf **Next.**
2. Lasse den eingestellten Pfad so, wie er ist, notiere Dir aber den Namen des Installationpfades (vermutlich *C:\User\DeinBenutzername\AppData\Local\Python35-32)*. Klicke auf **Next.**

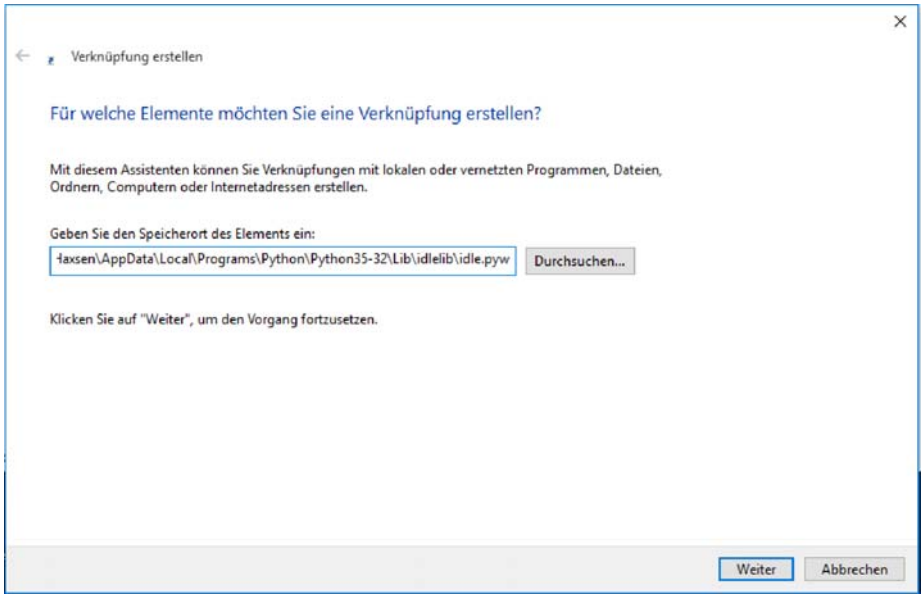
Am Ende dieses Vorgangs solltest Du einen Python-3-Eintrag in Deinem Start-Menü haben:

Als Nächstes machst Du Folgendes, um Dir eine Verknüpfung auf dem Desktop anzulegen:

1. Mache einen Rechtsklick auf Deinem Desktop, und wähle im Kontextmenü **Neu ▶ Verknüpfung**.
2. Im nun folgenden Dialogfenster, wo es heißt **Geben Sie den Speicherort des Elementes ein** (achte darauf, dass der Pfad der gleiche ist, den Du vorher notiert hast) gibst Du Folgendes ein:

   ```
   C:\Users\DeinBenutzername\AppData\Local\Programs\Python\
           Python35-32\Lib\idlelib\idle.pyw
   ```

   Das Dialogfenster sollte jetzt so aussehen:



3. Klicke auf **Weiter**, um zum nächsten Dialogfenster zu gelangen.
4. Als Namen gibst Du *IDLE* ein und klickst auf **Fertig stellen**, um die Verknüpfung zu erstellen.

Jetzt kannst Du zu »Wenn Du Python installiert hast« auf Seite 14 weiterblättern.

Python in MacOSX installieren

Falls Du einen Mac benutzt, solltest Du bereits eine Version von Python vorfinden. Dabei handelt es sich aber wahrscheinlich um eine ältere Version. Um ganz sicherzugehen, dass Du die aktuelle Version hast, gehst Du mit Deinem Browser auf *http://www.python.org/getit/* und lädst Dir den aktuellen Installer für Mac herunter.

Es gibt dort zwei verschiedene Installer. Welchen Du herunterladen solltest, hängt von der MacOSX-Version ab, die Du benutzt (um das herauszufinden, klickst Du in der obersten Menüleiste auf das Apple-Symbol und gehst auf **Über diesen Mac**.) Wähle dann wie folgt den Installer:

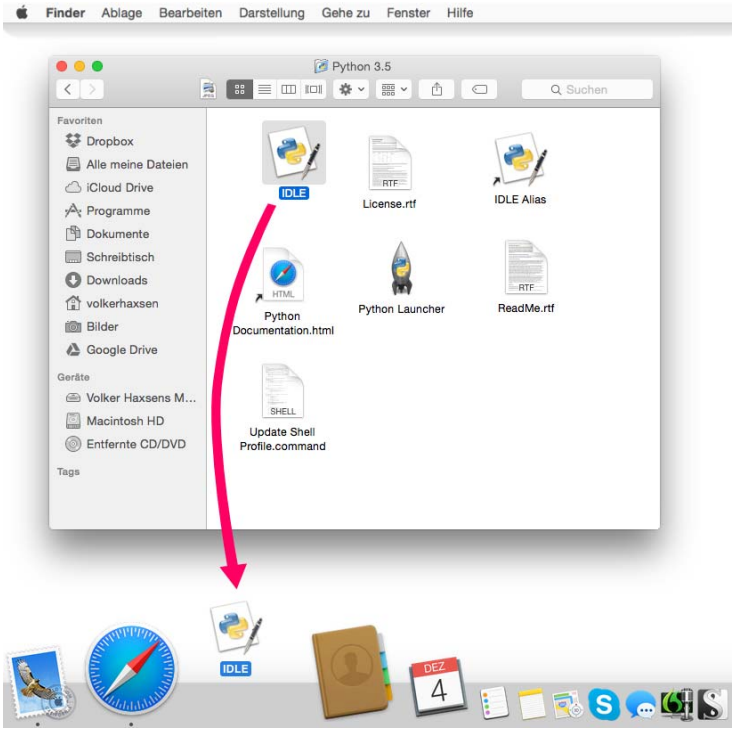
- Wenn Du eine MacOSX-Version zwischen 10.3 und 10.6 hast, lädst Du die 32-Bit-Version von Python 3 für i386/PPC herunter.
- Wenn Du die MacOSX-Version 10.6 oder eine höhere hast, lädst Du die 64-Bit/32-Bit-Version von Python 3 für x86-64/i386 herunter.

Sobald Du die Datei heruntergeladen hast (sie wird das Suffix *.pkg* haben), machst Du einen Doppelklick darauf. Danach siehst Du ein Fenster mit den Inhalten dieser Datei.

Anschließend folgst Du den Anweisungen beim Installieren der Software. Du wirst aufgefordert, Dein Administrator-Kennwort einzugeben, bevor sich Python installiert. (Du hast kein Administrator-Kennwort? Dann müssen es vielleicht Deine Eltern eingeben.)

Um Python schnell starten zu können, solltest Du dir das Icon von IDLE ins Dock legen:

1. Klicke in einem offenen Fenster auf **Programme**.
2. Gehe in der Liste der Programme bis zum Ordner **Python 3.5** und öffne ihn mit einem Doppelklick.
3. Klicke nun einmal auf das Icon »IDLE« und ziehe es mit gedrückter linker Maustaste (oder über das Trackpad) in Dein Dock. Hier ist es am unteren Bildschirmrand.
4. Von nun an kannst Du die Python-Shell direkt durch Anklicken aus Deinem Dock starten.

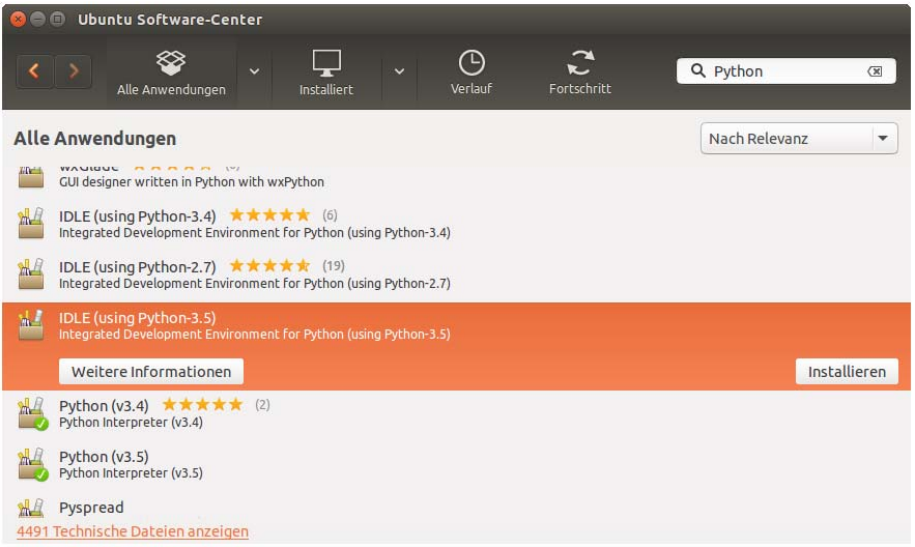


Jetzt kannst Du zu Seite 14, »Wenn Du Python installiert hast«, gehen und mit Python loslegen.

Python in Ubuntu installieren

Python ist bei der Ubuntu-Distribution schon vorinstalliert, aber es könnte sich dabei um eine ältere Version handeln. Um Python 3 in Ubuntu 15.x zu installieren, führe folgende Schritte durch:

1. Klicke in der Seitenleiste auf den Button für das Ubuntu-Software-Center. (Das ist das Icon, das wie eine orangefarbene Tasche aussieht – falls Du es nicht siehst, kannst Du auch auf den Dash-Startseite-Button klicken und in das Suchfeld *Software* eingeben.)
2. Gib im Suchfeld ganz oben rechts im Software-Center *Python ein.*
3. In der Liste der angebotenen Software wählst Du die aktuelle Version von IDLE, also in diesem Fall *IDLE (using Python 3.5),* aus.
4. Klicke auf **Installieren**.
5. Um die Software zu installieren, gibst Du Dein Administrator-Passwort ein und klickst dann auf **Authentifizieren**. (Du hast kein Administrator-Kennwort? Dann müssen es vielleicht Deine Eltern eingeben.)



Achtung!

Bei einigen Ubuntu-Versionen siehst Du vielleicht nur Python (v.3.5) im Hauptmenü (statt IDLE). Dies kannst Du dann stattdessen installieren.

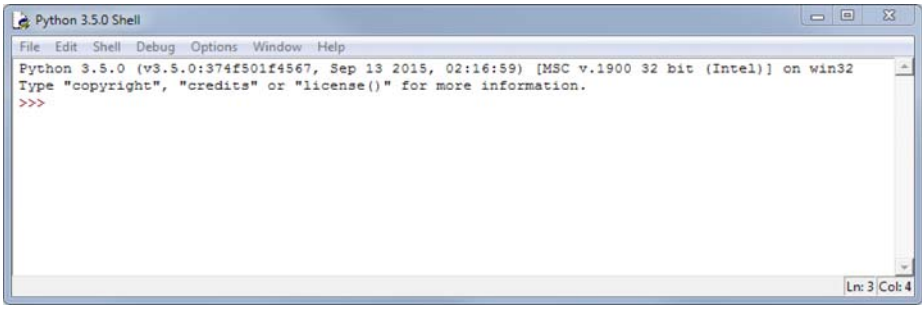
Jetzt, da Du die aktuelle Version von Python installiert hast, wollen wir es einmal ausprobieren.

2.3 Wenn Du Python installiert hast

Jetzt solltest Du ein Icon auf Deinem Windows- oder MacOSX-Schreibtisch respektive Desktop haben, das mit **IDLE** beschriftet ist. Wenn Du Ubuntu nutzt, solltest Du auf der **Dash-Startseite** unter *Anwendungen* **IDLE (using Python 3.5)** (oder eine spätere Version) finden.

Mache einen Doppelklick auf das Icon, oder wähle die Menü-Option. Danach sollte dieses Fenster erscheinen:

Dies ist die Python-Shell, die zur integrierten Enwicklungsumgebung von Python gehört. Die drei Größer-als-Zeichen (>>>) nennt man den *Prompt*.



Lasst uns nun einige Befehle hinter dem Prompt eingeben. Wir fangen mit diesem hier an:

```
>>> print("Hallo Welt!")
```

Achte darauf, dass Du die beiden Anführungsstriche oben (" ") mit eingibst. Drücke dann auf die ENTER-Taste auf Deiner Tastatur. Wenn Du den Befehl korrekt eingegeben hast, solltest Du so etwas sehen:

```
>>> print("Hallo Welt!")
Hallo Welt!
>>>
```

Der Prompt sollte danach wieder erscheinen, damit Du weißt, dass Python wieder bereit ist, weitere Befehle zu empfangen.

Glückwunsch! Du hast soeben Dein erstes Python-Programm geschrieben. Das Wort `print` gehört zu der Gruppe von Python-Befehlen, die man *Funktionen* nennt. Die Funktion `print` gibt alles auf dem Bildschirm aus, was zwischen den Anführungsstrichen steht. Du hast also dem Computer gesagt, dass er die Worte »Hallo Welt!« anzeigen soll – eine Anweisung also, die Du genauso verstehst wie auch der Computer.

2.4 Deine Python-Programme sichern

Python-Programme wären nicht sehr nützlich, wenn man sie jedes Mal wieder neu schreiben müsste, wenn man sie benutzen möchte. Außerdem müsste man sie auch noch ausdrucken, um eine Vorlage fürs nächste Mal zu haben.

Natürlich ist es kein Problem, kleine Programme neu zu schreiben, aber große Programme, wie etwa eine Textverarbeitung, können Millionen von Programmzeilen enthalten. Wenn Du die ausdrucken würdest, hättest Du weit über