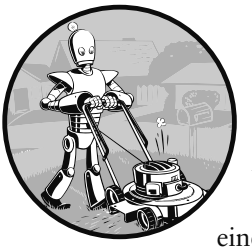
Einleitung



»Du hast gerade in zwei Stunden das erledigt, woran wir drei sonst zwei Tage lang sitzen!«

Mein Mitbewohner in den frühen 2000er-Jahren arbeitete bei einem Elektronikhändler, bei dem gelegentlich eine Tabelle mit Tausenden von Produktpreisen anderer Läden auftauchte. Drei Mitarbeiter druckten diese Tabelle dann auf einem dicken Stapel Papier aus und teilten diesen unter sich auf. Für jeden Produktpreis schlugen sie den Preis ihres eigenen Arbeitgebers nach und notierten alle Produkte, die die Konkurrenz billiger anbot. Damit waren sie gewöhnlich zwei Tage beschäftigt.

»Wenn ich die Originaldatei bekomme, kann ich ein Programm schreiben, das die Arbeit für euch erledigt«, schlug mein Mitbewohner ihnen vor, als er sah, wie sie inmitten eines Riesenhaufens Papier auf dem Fußboden hockten.

Nach ein paar Stunden hatte er ein kurzes Programm geschrieben, das die Preisliste der Konkurrenten aus der Datei auslas, die Produkte in der Datenbank des Elektronikladens nachschlug und einen Vermerk machte, wenn die Konkurrenz billiger war. Er war immer noch ein Anfänger in Sachen Programmierung

und hatte den Großteil dieser Stunden damit zugebracht, die Dokumentation in einem Programmierbuch nachzuschlagen. Die Ausführung des fertigen Programms dauerte nur wenige Sekunden. An dem Tag gönnten sich mein Mitbewohner und seine Kollegen eine besonders lange Mittagspause.

Das zeigt das Potenzial der Programmierung. Ein Computer ist wie ein Schweizer Messer und lässt sich für zahllose Aufgaben einrichten. Viele Leute bringen Stunden mit Klicken und Tippen zu, um monotone Aufgaben auszuführen, ohne zu ahnen, dass der Computer diese Arbeit in wenigen Sekunden erledigen könnte, wenn er nur die richtigen Anweisungen dafür bekäme.

Für wen ist dieses Buch gedacht?

Software bildet die Grundlage vieler unserer Geräte, die wir für die Arbeit und in der Freizeit verwenden: Fast jeder nutzt soziale Netzwerke zur Kommunikation, die Telefone vieler Menschen enthalten Computer mit Internetzugriff und für die meisten Büroarbeiten ist Computerarbeit erforderlich. Daher ist die Nachfrage nach Personen, die programmieren können, sprunghaft angestiegen. Unzählige Bücher, interaktive Webtutorials und Schulungen für Entwickler werden mit dem Versprechen beworben, ehrgeizige Anfänger zu Softwareingenieuren zu machen, die sechsstellige Gehälter verlangen können.

Dieses Buch ist jedoch nicht für diese Leute gedacht, sondern für alle anderen.

Mit diesem Buch allein können Sie nicht zu einem professionellen Softwareentwickler werden, genauso wenig, wie ein paar Gitarrenstunden Sie zu einem Rockstar machen. Wenn Sie aber Büroangestellter, Administrator oder Akademiker sind oder auch nur zur Arbeit oder zum Vergnügen einen Computer benutzen, so werden Sie hier die Grundlagen der Programmierung kennenlernen, um einfache Aufgaben wie die folgenden zu automatisieren:

- Tausende von Dateien verschieben und umbenennen und in Ordner sortieren
- Onlineformulare ausfüllen, ohne Text eingeben zu müssen
- Dateien von einer Website herunterladen oder Texte von dort kopieren, sobald dort neues Material bereitgestellt wird
- Sich von Ihrem Computer per SMS benachrichtigen lassen
- Excel-Arbeitsblätter bearbeiten und formatieren
- Nach neuen E-Mails suchen und vorformulierte Antworten senden

Diese Aufgaben sind einfach, aber zeitraubend, und sie sind häufig so trivial oder so spezifisch, dass es keine fertige Software dafür gibt. Mit einigen Programmierkenntnissen können Sie Ihren Computer diese Aufgaben für Sie erledigen lassen.

Programmierstil

Dieses Buch ist nicht als Nachschlagewerk gedacht, sondern als Anleitung für Anfänger. Der Programmierstil verstößt manchmal gegen die üblichen Richtlinien (beispielsweise werden in einigen Programmen globale Variablen verwendet), aber das ist ein Kompromiss, um das Lernen zu erleichtern. In diesem Buch lernen Sie, Wegwerfcode für einmalige Aufgaben zu schreiben, weshalb wir nicht viel Mühe auf Stil und Eleganz verwenden. Auch anspruchsvolle Programmierkonzepte wie Objektorientierung, Listenabstraktion und Generatoren werden hier aufgrund ihrer Kompliziertheit nicht behandelt. Altgediente Programmierer werden den Code sicherlich ändern wollen, um die Effizienz zu erhöhen, aber in diesem Buch geht es darum, Programme mit so wenig Aufwand wie möglich zum Laufen zu bekommen.

Was ist Programmierung?

In Filmen und Fernsehserien werden Programmierer oft als Leute dargestellt, die rasend schnell auf einer Tastatur herumtippen, um kryptische Folgen von Nullen und Einsen auf leuchtenden Bildschirmen erscheinen zu lassen. In Wirklichkeit ist moderne Programmierung aber nicht so geheimnisvoll. *Programmierung* ist einfach die Eingabe von Anweisungen, die der Computer ausführen soll. Diese Anweisungen können dazu dienen, mit Zahlen zu rechnen, Text zu ändern, Informationen in Dateien nachzuschlagen oder über das Internet mit anderen Computern zu kommunizieren.

Alle Programme bestehen aus einfachen Anweisungen, die die Grundbausteine darstellen. Einige der gebräuchlichsten dieser Anweisungen besagen, auf Deutsch übersetzt, Folgendes:

»Mach dies; dann mach das.«

»Wenn diese Bedingung wahr ist, dann führe diese Aktion aus; anderenfalls jene Aktion.«

»Mach dies genau 27 Mal.«

»Mach dies, solange die Bedingung wahr ist.«

Diese Bausteine können Sie kombinieren, um auch kompliziertere Entscheidungen zu treffen. Im folgenden Beispiel sehen Sie die Programmieranweisungen – den *Quellcode* – für ein einfaches Programm in der Programmiersprache Python. Die Software Python führt die einzelnen Codezeilen vom Anfang bis zum Ende aus. (Manche Zeilen werden nur ausgeführt, wenn (*if*) eine Bedingung wahr ist (*true*); anderenfalls (*else*) führt Python eine andere Zeile aus.)

```
passwordFile = open('SecretPasswordFile.txt')   ❶
secretPassword = passwordFile.read()   ❷
print('Enter your password.')   ❸
typedPassword = input()
if typedPassword == secretPassword:   ❹
    print('Access granted')   ❺
    if typedPassword == '12345':   ❻
        print('That password is one that an idiot puts on their luggage.')  ❼
else:
    print('Access denied')   ❽
```

Auch wenn Sie noch nicht viel von Programmierung verstehen, können Sie vielleicht schon erraten, was der vorstehende Code bewirkt. Als Erstes wird die Datei *SecretPasswordFile.txt* geöffnet (❶) und das geheime Passwort gelesen (❷). Danach wird der Benutzer aufgefordert, ein Passwort einzugeben (über die Tastatur) (❸). Die beiden Passwörter werden verglichen (❹), und wenn sie identisch sind, gibt das Programm auf dem Bildschirm die Meldung *Access granted* (»Zugriff gewährt«) aus (❺). Danach prüft das Programm, ob das Passwort *12345* lautet (❻). Wenn ja, gibt es dem Benutzer den dezenten Hinweis, dass dies nicht gerade die ideale Wahl für ein Passwort ist (❼). Sind die Passwörter nicht identisch, gibt das Programm *Access denied* (»Zugriff verweigert«) aus (❽).

Was ist Python?

Der Begriff *Python* bezeichnet die Programmiersprache Python (deren Syntaxregeln festlegen, was als gültiger Python-Code angesehen wird) und den Python-Interpreter, eine Software, die den (in der Sprache Python geschriebenen) Code liest und dessen Anweisungen ausführt. Den Python-Interpreter können Sie kostenlos von *https://python.org/* herunterladen, wobei es Versionen für Linux, macOS und Windows gibt.

Der Name Python ist übrigens nicht von der Schlange abgeleitet, sondern von der surrealistischen britischen Komikergruppe Monty Python. Python-Programmierer werden liebevoll »Pythonistas« genannt, und Tutorials sowie die Dokumentation zu Python stecken voller Anspielungen sowohl auf Monty Python als auch auf Schlangen.

Programmierer müssen nicht viel Mathe können

Wenn mir jemand erklärt, warum er Angst davor hat, Programmieren zu lernen, geht es meistens darum, dass er glaubt, dazu müsste man sehr gut in Mathematik sein. In Wirklichkeit ist zur Programmierung meistens nicht mehr Mathe als einfache Grundrechenarten erforderlich. Programmieren lässt sich in diesem Punkt

sogar mit dem Lösen von Sudoku-Rätseln vergleichen. Dazu müssen Sie in jede Zeile, jede Spalte und jedes innere 3x3-Quadrat des 9x9-Feldes die Zahlen von 1 bis 9 einfügen, wobei bereits einige Zahlen vorgegeben sind. Aus diesen Zahlen leiten Sie die Lösung durch Deduktion und Logik ab. In der Aufgabe in Abb. ***E–1*** kommt beispielsweise eine 5 sowohl in der ersten als auch in der zweiten Zeile vor. Daher muss die 5 im oberen rechten Quadrat in der dritten Zeile stehen. Da sich auch schon eine 5 in der letzten Spalte befindet, kann die 5 in der dritten Zeile nicht rechts neben der 6 stehen. Damit bleibt nur der Platz links von der 6 übrig. Jede Zeile, jede Spalte und jedes Quadrat, das Sie lösen, gibt Ihnen weitere Hinweise für den Rest des Rätsels. Mit jeder Gruppe der Zahlen von 1 bis 9, die Sie vervollständigen, nähern Sie sich der Lösung des gesamten Rätsels.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Abb. 2–1 *Ein Sudoku-Rätsel (links) und die Lösung (rechts). Beim Sudoku geht es zwar um Zahlen, doch ist dafür keine Mathematik erforderlich. (Bilder (c) Wikimedia Commons)*

Nur weil es bei Sudoku um Zahlen geht, heißt das noch lange nicht, dass man gut in Mathe sein muss, um die Lösung auszuknobeln. Das Gleiche gilt auch fürs Programmieren. Wie beim Sudoku müssen Sie auch beim Programmieren das Problem in einzelne Schritte zerlegen. Beim *Debuggen* von Programmen (also beim Aufspüren und Beheben von Fehlern) müssen Sie geduldig beobachten, was das Programm macht, und die Ursachen von Fehlern herausfinden. Wie bei allen anderen Fähigkeiten werden Sie auch beim Programmieren umso besser, je mehr Erfahrung Sie haben.

Sie sind nie zu alt, um programmieren zu lernen

Den zweithäufigsten Einwand, den ich im Zusammenhang mit Programmierung zu hören bekomme, ist, dass jemand glaubt, zu alt zu sein, um diese Tätigkeit noch zu erlernen. Im Internet musste ich schon viele Kommentare von Leuten lesen, die

meinten, sie seien mit sage und schreibe 23 Jahren (!) schon zu alt dafür. Das ist ganz sicher nicht »zu alt«, um programmieren zu lernen. Viele Menschen eignen sich in viel höherem Alter neue Fähigkeiten an.

Man muss nicht als Kind anfangen, um ein fähiger Programmierer zu werden. Die Vorstellung von Programmierern als Wunderkindern ist aber nicht totzukriegen. Leider habe ich selbst zu der Legende beigetragen, weil ich erzählt habe, dass ich bereits in der Grundschule mit dem Programmieren angefangen habe.

Programmieren lässt sich heutzutage leichter lernen als in den 90ern, denn heute gibt es viel mehr Bücher, bessere Suchmaschinen und viel mehr Websites, auf denen Sie Antworten auf Ihre Fragen bekommen. Vor allem aber sind die Programmiersprachen selbst viel benutzerfreundlicher geworden. Daher können Sie sich heute alles, was ich zwischen Grundschule und High-School-Abschluss über Programmierung gelernt habe, an ungefähr einem Dutzend Wochenenden erarbeiten. Mein Vorsprung war in Wirklichkeit gar kein so großer Vorsprung.

Programmieren lernt man nur durch Übung. Wir werden nicht als Programmierer geboren, und noch nicht programmieren zu können, heißt nicht, dass man niemals ein Experte auf diesem Gebiet werden könnte.

Programmierung ist kreativ

Programmieren ist eine ebenso kreative Tätigkeit wie Malen, Schreiben, Stricken und das Bauen mit Lego-Steinen. Genauso wie das Malen auf einer leeren Leinwand ist die Entwicklung von Software zwar gewissen Einschränkungen unterworfen, bietet aber auch unendlich viele Möglichkeiten.

Der Unterschied zwischen Programmierung und anderen kreativen Tätigkeiten besteht darin, dass Sie beim Programmieren das gesamte erforderliche Rohmaterial in Ihrem Computer haben. Sie müssen keine Leinwand, keine Farbe, keinen Film, kein Garn, keine Lego-Steine oder elektronischen Bauteile kaufen. Ein zehn Jahre alter Computer ist mehr als leistungsfähig genug, um damit Programme zu schreiben. Ein fertiges Programm können Sie beliebig oft kopieren. Während ein gestrickter Pullover immer nur von einer einzigen Person auf einmal getragen werden kann, lässt sich ein nützliches Programm auf einfache Weise der ganzen Welt online zur Verfügung stellen.

Der Aufbau dieses Buchs

Der erste Teil dieses Buchs behandelt die Grundlagen der Python-Programmierung. Im zweiten Teil sehen wir uns dann verschiedene Aufgaben an, die Sie automatisieren können. In jedem Kapitel des zweiten Teils gibt es Übungsprojekte. Die folgende Übersicht zeigt, was Sie in den einzelnen Kapiteln erwartet:

Teil I: Grundlagen der Python-Programmierung

Kapitel 1: Grundlagen von Python Hier werden Ausdrücke vorgestellt, die grundlegendste Art von Python-Anweisungen. Außerdem erfahren Sie, wie Sie die interaktive Shell von Python verwenden, um Code auszuprobieren.

Kapitel 2: Flusssteuerung In diesem Kapitel erfahren Sie, wie Ihre Programme entscheiden können, welcher Code in einer bestimmten Situation ausgeführt werden soll. Dadurch können Sie auf unterschiedliche Bedingungen reagieren.

Kapitel 3: Funktionen Dieses Kapitel zeigt Ihnen, wie Sie eigene Funktionen definieren, um Ihren Code in besser handhabbare Abschnitte zu gliedern.

Kapitel 4: Listen Hier erhalten Sie eine Einführung in den Datentyp der Listen und erfahren, wie Sie damit Daten gliedern können.

Kapitel 5: Dictionaries und Datenstrukturen Dieses Kapitel gibt eine Einführung in den Datentyp der Dictionaries und führt noch weitere Möglichkeiten auf, um Daten zu gliedern.

Kapitel 6: Stringbearbeitung Hier geht es um die Arbeit mit Textdaten (die in Python *Strings* genannt werden).

Teil II: Aufgaben automatisieren

Kapitel 7: Mustervergleich mit regulären Ausdrücken Hier erfahren Sie, wie Sie mit regulären Ausdrücken nach Textmustern suchen können.

Kapitel 8: Eingabevalidierung Hier wird erläutert, wie Ihr Programm die von Benutzern eingegebenen Informationen überprüfen kann, damit die Daten nicht in einer Form vorliegen, die im weiteren Verlauf des Programms Fehler verursachen kann.

Kapitel 9: Dateien lesen und schreiben Dieses Kapitel erklärt, wie Ihre Programme den Inhalt von Textdateien lesen und selbst Informationen in Dateien auf der Festplatte speichern können.

Kapitel 10: Dateien verwalten Sie erfahren hier, wie Python große Mengen von Dateien kopieren, verschieben, umbenennen und löschen kann, und zwar viel schneller, als ein menschlicher Bearbeiter es tun könnte. Außerdem werden das Komprimieren und Entpacken von Dateien erklärt.

Kapitel 11: Debugging Hier werden die verschiedenen Instrumente vorgestellt, die in Python zur Verfügung stehen, um Fehler (Bugs) zu finden und zu beheben.

Kapitel 12: Web Scraping Dieses Kapitel zeigt Ihnen, wie Sie Programme schreiben, die automatisch Webseiten herunterladen und nach Informationen durchforsten. Dieser Vorgang wird *Web Scraping* genannt.

Kapitel 13: Excel-Arbeitsblätter Hier geht es darum, wie Sie Excel-Arbeitsblätter programmgesteuert bearbeiten, sodass Sie das nicht manuell tun müssen. Das ist besonders praktisch, wenn die Anzahl der Dokumente, die Sie analysieren müssen, in die Hunderte oder gar in die Tausende geht.

Kapitel 14: Google Tabellen In diesem Kapitel erfahren Sie, wie Sie mit Python Daten in der häufig genutzten Online-Tabellenkalkulationsanwendung Google Tabellen lesen und bearbeiten.

Kapitel 15: PDF- und Word-Dokumente Dieses Kapitel behandelt das programmgesteuerte Lesen von Word- und PDF-Dokumenten.

Kapitel 16: CSV-Dateien und JSON-Daten Die Erklärung der programmgesteuerten Bearbeitung von Dokumenten wird hier für CSV- und JSON-Dateien fortgesetzt.

Kapitel 17: Zeit und Aufgabenplanung Hier lernen Sie, wie Python Uhrzeiten und Kalenderdaten handhabt und wie Sie dafür sorgen, dass Ihr Computer Aufgaben zu einem bestimmten Zeitpunkt ausführt. Außerdem erfahren Sie, wie Sie von Python-Programmen aus andere Programme starten.

Kapitel 18: E-Mails und Textnachrichten In diesem Kapitel geht es darum, Programme zu schreiben, die an Ihrer Stelle E-Mails und Textnachrichten senden.

Kapitel 19: Bildbearbeitung Dieses Kapitel erklärt, wie Sie Bilder, z. B. JPEG- oder PNG-Dateien, in Ihren Programmen bearbeiten können.

Kapitel 20: GUI-Automatisierung Hier lernen Sie, wie Sie mit einem Programm die Maus und die Tastatur steuern, um Mausklicks und Tastenbetätigungen zu simulieren.

Anhang A: Drittanbietermodule installieren Dieser Anhang zeigt, wie Sie Python mithilfe von nützlichen Modulen erweitern können.

Anhang B: Programme ausführen Hier erfahren Sie, wie Sie Python-Programme unter Windows, macOS und Linux außerhalb des Codeeditors ausführen.

Anhang C: Antworten auf die Wiederholungsfragen Hier finden Sie die Lösungen sowie einige zusätzliche Erklärungen zu den Wiederholungsfragen, die am Ende jedes Kapitels stehen.

Python herunterladen und installieren

Python können Sie kostenlos für Windows, macOS und Linux von *https://python.org/downloads/* herunterladen. Wenn Sie die neueste Version verwenden, die auf der Website angeboten wird, sollten alle Programme in diesem Buch funktionieren.

Warnung

Achten Sie darauf, eine Version von Python 3 herunterzuladen (z. B. 3.8.0). Die Programme in diesem Buch sind für Python 3 geschrieben. Auf Python 2 funktionieren sie unter Umständen gar nicht oder zumindest nicht korrekt.

Auf der Downloadseite finden Sie Installer für die verschiedenen Betriebssysteme und dabei wiederum jeweils für 64- und für 32-Bit-Computer. Als Erstes müssen Sie daher herausfinden, welchen Installer Sie brauchen. Wenn Sie Ihren Computer 2007 oder später gekauft haben, handelt es sich sehr wahrscheinlich um ein 64-Bit-System, anderenfalls eher um einen 32-Bit-Rechner. Genau herausfinden können Sie das wie folgt:

- Auf Windows wählen Sie *Start* > *Systemsteuerung* > *System* und schauen nach, ob als Systemtyp 64 Bit oder 32 Bit angegeben wird.
- Auf macOS wählen Sie im Apfelmenü *Über diesen Mac* > *Weitere Informationen* > *Systembericht* > *Hardware*. Schauen Sie sich in der *Hardware-Übersicht* den Eintrag unter *Prozessortyp* an. Wenn dort *Intel Core Solo* oder *Intel Core Duo* steht, haben Sie einen 32-Bit-Rechner. Bei allen anderen Einträgen (auch *Intel Core 2 Duo*) handelt es sich um einen 64-Bit-Computer.
- Auf Ubuntu Linux geben Sie in einem Terminalfenster den Befehl `uname -m` ein. Die Antwort `i686` bedeutet, dass Sie einen 32-Bit-Computer haben. Bei einem 64-Bit-Rechner lautet die Antwort `x86_64`.

Laden Sie auf Windows den Python-Installer (mit der Endung *.msi*) herunter und doppelklicken Sie darauf. Befolgen Sie die Anweisungen, die auf dem Bildschirm angezeigt werden. Der Vorgang läuft wie folgt ab:

1. Wählen Sie *Install for All Users* und dann *Weiter*.
2. Akzeptieren Sie in den nächsten Fenstern jeweils die Standardoptionen, indem Sie auf *Weiter* klicken.

Auf macOS laden Sie die passende *.dmg*-Datei für Ihre Version des Betriebssystems herunter und doppelklicken darauf. Befolgen Sie die Anweisungen, die auf dem Bildschirm angezeigt werden. Der Vorgang läuft wie folgt ab:

1. Wenn das DMG-Paket in einem neuen Fenster geöffnet wird, doppelklicken Sie auf die Datei *Python.mpkg*. Möglicherweise müssen Sie Ihr Administratorpasswort eingeben.
2. Akzeptieren Sie in den nächsten Fenstern jeweils die Standardoptionen, indem Sie auf *Weiter* klicken, und klicken Sie auf *Agree*, um die Lizenzbedingungen zu akzeptieren.
3. Klicken Sie im letzten Fenster auf *Install*.

Auf Ubuntu können Sie Python wie folgt im Terminal installieren:

1. Öffnen Sie ein Terminalfenster.
2. Geben Sie `sudo apt-get install python3` ein.
3. Geben Sie `sudo apt-get install idle3` ein.
4. Geben Sie `sudo apt-get install python3-pip` ein.

Mu herunterladen und installieren

Der *Python-Interpreter* ist die Software, die Ihre Python-Programme ausführt. Die Eingabe der Programme dagegen erfolgt im *Editor Mu* ähnlich wie in einer Textverarbeitung. Mu können Sie von *https://codewith.mu/* herunterladen.

Auf Windows und macOS laden Sie die Installerdatei für Ihr Betriebssystem herunter und führen sie aus, indem Sie darauf doppelklicken. Auf macOS wird dadurch ein Fenster geöffnet, in dem Sie das Mu-Symbol in den Ordner *Programm* ziehen müssen, um mit der Installation fortzufahren. Auf Ubuntu installieren Sie Mu als Python-Paket. Klicken Sie dazu auf der Downloadseite im Abschnitt *Python Package* auf die Schaltfläche *Instructions*.

Mu starten

Um Mu zu starten, gehen Sie wie folgt vor:

- Auf Windows 7 und höher klicken Sie auf das Startsymbol in der linken unteren Ecke, geben **Mu** in das Suchfeld ein und wählen die Anwendung aus.
- Auf macOS öffnen Sie ein Finder-Fenster, klicken auf *Programme* und dann auf *mu-editor*.
- Auf Ubuntu wählen Sie *Anwendungen* > *Zubehör* > *Terminal* und geben dann **python3 -m mu** ein.

Wenn Sie Mu zum ersten Mal ausführen, erscheint das Fenster *Select Mode* mit den Optionen *Adafruit CircuitPython*, *BBC micro:bit*, *Pygame Zero* und *Python 3*. Wählen Sie hier *Python 3*. Bei Bedarf können Sie den Modus später wieder ändern. Klicken Sie dazu auf die Schaltfläche *Mode* oben im Editorfenster.

Hinweis

Um die in diesem Buch verwendeten Drittanbietermodule installieren zu können, brauchen Sie Mu in der Version 1.1.0. Sie steht zurzeit als Alpha-Release unter einem eigenen Link getrennt von den anderen Downloadlinks zur Verfügung.

IDLE starten

In diesem Buch verwenden wir Mu als Editor und als interaktive Shell. Um Python-Code zu schreiben, können Sie jedoch auch beliebige andere Editoren verwenden. Zusammen mit Python wird die Software *IDLE* (*Integrated Development and Learning Environment*) installiert, die Sie ebenfalls als Editor verwenden können, wenn Sie Mu aus irgendeinem Grunde nicht installieren oder zum Laufen bekommen können. Um IDLE zu starten, gehen Sie wie folgt vor:

- Auf Windows 7 und höher klicken Sie auf das Startsymbol in der linken unteren Ecke, geben **IDLE** in das Suchfeld ein und wählen *IDLE (Python GUI)* aus.
- Auf macOS öffnen Sie ein Finder-Fenster, klicken auf *Programme*, dann auf *Python 3.8* und schließlich auf das IDLE-Symbol.
- Auf Ubuntu wählen Sie *Anwendungen* > *Zubehör* > *Terminal* und geben dann **idle3** ein. (Möglicherweise können Sie auch oben auf dem Bildschirm auf *Anwendungen* klicken, dann *Programming* auswählen und auf *IDLE3* klicken.)

Die interaktive Shell

Wenn Sie Mu ausführen, sehen Sie normalerweise das *Editorfenster*. Sie können in Mu allerdings auch eine *interaktive Shell* öffnen, indem Sie auf die Schaltfläche *REPL* klicken. Eine Shell ist ein Programm, in das Sie Anweisungen für den Computer eingeben können, ähnlich wie im Terminal von macOS oder an der Eingabeaufforderung von Windows. In die interaktive Python-Shell geben Sie die Anweisungen ein, die der Python-Interpreter ausführen soll. Der Computer liest diese Anweisungen und setzt sie sofort um.

In Mu ist die interaktive Shell ein Bereich in der unteren Hälfte des Fensters. Dort wird folgender Text angezeigt:

```
Jupyter QtConsole 4.3.1
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 18:11:49) [MSC v.1900 64 bit
(AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

Wenn Sie IDLE verwenden, erscheint als erstes Fenster die interaktive Shell. Sie ist bis auf den folgenden Text größtenteils leer:

```
Python 3.8.0b1 (tags/v3.8.0b1:3b5deb0116, Jun 4 2019, 19:52:55) [MSC v.1916
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Bei `[1]:` und `>>>` handelt es sich jeweils um die *Eingabeaufforderung*. In diesem Buch verwenden wir dafür das Zeichen `>>>`, da es gebräuchlicher ist. Auch im Terminal oder an der Windows-Eingabeaufforderung sehen Sie `>>>`. Die Eingabeaufforderung `[1]:` wurde für Jupyter Notebook erfunden, einen weiteren weitverbreiteten Python-Editor.

Zum Ausprobieren geben Sie an der Eingabeaufforderung der Shell Folgendes ein:

```
>>> print('Hello, world!')
```

Wenn Sie nun die Eingabetaste drücken, zeigt die interaktive Shell die Reaktion an:

```
>>> print('Hello, world!')
Hello, world!
```

Damit haben Sie Ihrem Computer gerade eine Anweisung erteilt, und er hat genau das getan, was Sie von ihm verlangt haben!

Drittanbietermodule installieren

Für manche Zwecke muss ein Programm Module importieren. Einige davon sind im Lieferumfang von Python enthalten, aber andere – sogenannte Drittanbietermodule – wurden von Programmierern erstellt, die nicht zum Hauptentwicklerteam von Python gehören. In Anhang A finden Sie eine ausführliche Anleitung, um solche Module mit dem Programm `pip` (auf Windows) bzw. `pip3` (auf macOS und Linux) zu installieren. Wenn Sie in diesem Buch angewiesen werden, ein bestimmtes Drittanbietermodul zu installieren, schlagen Sie in Anhang A nach.

Hilfe finden

Programmierer neigen dazu, im Internet nach Antworten auf ihre Fragen zu suchen. Das ist eine ganz andere Art des Lernens, als viele es gewohnt sind. Es gibt hier keinen persönlich anwesenden Lehrer, der Ihnen etwas beibringt und Ihre Fragen beantwortet. Was das Internet als Klassenzimmer auszeichnet, ist die Tatsache, dass es dort sehr viele Menschen gibt, die Ihre Fragen beantworten können. Höchstwahrscheinlich ist Ihre Frage auch schon längst beantwortet worden, sodass die Lösung lediglich darauf wartet, dass Sie sie finden. Wenn Sie eine Fehlermeldung erhalten oder Schwierigkeiten haben, den Code das machen zu lassen, was er soll, sind Sie nicht die erste Person, die sich diesem Problem gegenübersieht. Daher ist es viel einfacher, als Sie glauben, eine Lösung zu finden.

Um Ihnen ein Beispiel zu geben, provozieren wir absichtlich einen Fehler: Geben Sie in die interaktive Shell **'42' + 3** ein. Machen Sie sich keine Gedanken darüber, was diese Anweisung bedeutet und was daran falsch sein soll, sondern achten Sie auf das Ergebnis:

```
>>> '42' + 3
Traceback (most recent call last):   ❶
  File "<pyshell#0>", line 1, in <module>
    '42' + 3
TypeError: Can't convert 'int' object to str implicitly   ❷
>>>
```

Da Python die Anweisung nicht versteht, erscheint hier eine Fehlermeldung (❷). Der als »Traceback« bezeichnete Teil der Fehlermeldung (❶) gibt die Anweisung und die Nummer der Zeile an, mit der Python Schwierigkeiten hat. Wenn Sie eine Fehlermeldung erhalten, die Ihnen schleierhaft ist, suchen Sie online danach. In diesem Fall also würden Sie **"TypeError: Can't convert 'int' object to str implicitly"** (in Anführungszeichen) in eine Suchmaschine eingeben. Daraufhin sehen Sie haufenweise Links, in denen erklärt wird, was diese Fehlermeldung bedeutet und was die Ursache ist (siehe ***E–2***).

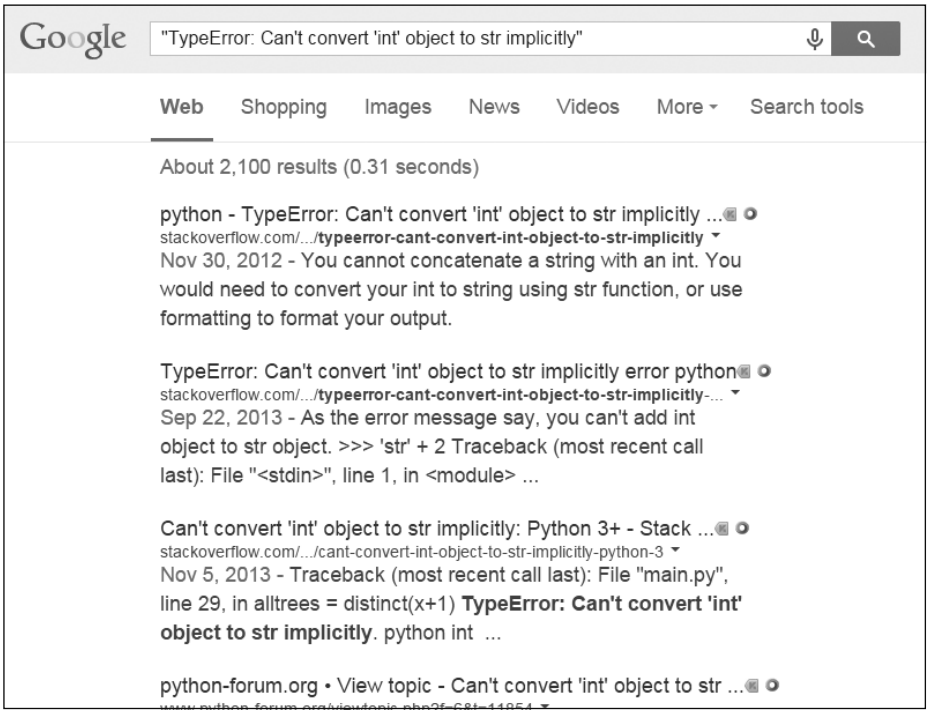


Abb. 2–2 *Google-Ergebnisse zu einer Fehlermeldung können sehr hilfreich sein.*

Sie werden dabei sehr oft bemerken, dass schon einmal jemand die gleiche Frage gestellt hat wie Sie und irgendeine hilfsbereite Person sie bereits beantwortet hat. Niemand kann alles über Programmierung wissen. Zur täglichen Arbeit eines Softwareentwicklers gehört auch die Suche nach Antworten auf technische Fragen.

Sinnvolle Fragen stellen

Wenn Sie die Antworten auf Ihre Fragen nicht durch eine Onlinesuche finden können, versuchen Sie, Teilnehmer in Webforen wie Stack Overflow (*https://stackoverflow.com/*) oder dem Subreddit »Learn Programming« auf *https://reddit.com/r/learnprogramming/* zu fragen. Beachten Sie aber, dass Sie Ihre Fragen geschickt stellen müssen, damit andere Ihnen helfen können. Lesen Sie auf jeden Fall den FAQ-Abschnitt der Website, um zu erfahren, wie Sie Fragen auf richtige Weise vorbringen.

Wenn Sie Fragen zur Programmierung stellen, sollten Sie Folgendes tun:

- Erklären Sie nicht nur, was Sie getan haben, sondern auch, was Sie tun wollten. Dadurch können Helfer erkennen, ob Sie sich verrannt haben.

- Geben Sie genau an, wann der Fehler auftritt. Zeigt er sich gleich zu Beginn des Programms oder erst nach einer bestimmten Aktion?
- Kopieren Sie die *gesamte* Fehlermeldung und Ihren Code auf *https://pastebin.com/* oder *https://gist.github.com/*. Diese Websites erleichtern es, anderen Personen große Mengen an Code über das Web zur Verfügung zu stellen, ohne die Formatierung zu verlieren. Die URL zu dem dort veröffentlichten Code fügen Sie dann in Ihre E-Mail oder Ihren Forumspost ein. Als Beispiele können Sie sich Code von mir auf *https://pastebin.com/SzP2DbFx/* und *httpss://gist.github.com/asweigart/6912168/* ansehen.
- Erklären Sie, was Sie bereits versucht haben, um das Problem zu lösen. Das zeigt den anderen, dass Sie selbst schon etwas Mühe darin investiert haben, die Lösung herauszufinden.
- Geben Sie an, welche Version von Python Sie verwenden. (Es gibt einige entscheidende Unterschiede zwischen den Python-Interpretern der Versionen 2 und 3.) Nennen Sie auch die Version Ihres Betriebssystems.
- Wenn ein Fehler nach einer Änderung am Code auftrat, erklären Sie, was Sie genau geändert haben.
- Geben Sie an, ob der Fehler jedes Mal auftritt, wenn Sie das Programm ausführen, oder nur, nachdem Sie bestimmte Aktionen durchgeführt haben. Beschreiben Sie in letzterem Fall auch diese Aktionen.

Befolgen Sie immer die Online-Etikette. Schreiben Sie also Ihre Posts nicht komplett in Großbuchstaben und stellen Sie keine unsinnigen Forderungen an die Menschen, die Ihnen zu helfen versuchen.

Weitere Informationen darüber, wie Sie um Hilfe bei Programmierfragen bitten können, erhalten Sie in dem Blogpost auf *https://autbor.com/help/*. Eine Liste häufig gestellter Fragen über Programmierung finden Sie auf *https://www.reddit.com/r/learnprogramming/wiki/faq/*. Eine ähnliche Liste zu dem Thema, einen Job im Bereich der Softwareentwicklung zu bekommen, steht auf *https://www.reddit.com/r/cscareerquestions/wiki/index/*.

Ich helfe anderen gern dabei, Python kennenzulernen. So schreibe ich Programmiertutorials in meinem Blog auf *https://inventwithpython.com/blog/*. Sie können sich mit Ihren Fragen auch über *al@inventwithpython.com* an mich wenden (in Englisch). Eine schnellere Antwort dürften Sie allerdings erhalten, wenn Sie Ihre Fragen auf *https://reddit.com/r/inventwithpython/* stellen.

Zusammenfassung

Für die meisten Menschen ist ein Computer eher ein Gerät als ein Werkzeug. Wenn Sie jedoch zu programmieren lernen, steht Ihnen eines der vielseitigsten Werkzeuge der modernen Welt zur Verfügung, und obendrein werden Sie auch noch Spaß dabei haben. Programmierung ist keine Gehirnchirurgie – Sie können sich auch als Anfänger daran versuchen und gefahrlos Fehler machen.

Für dieses Buch müssen Sie keinerlei Programmierkenntnisse mitbringen. Es kann aber sein, dass Sie Fragen haben, die über den behandelten Stoff hinausgehen. Die richtigen Fragen zu stellen und zu wissen, wo Sie Antworten finden können, ist für Programmierer von unschätzbarem Wert.

Fangen wir an!