Inhalt (im Überblick)

Inhalt (jetzt ausführlich)

Einführung

Ihr Gehirn und das Programmieren. *Sie* versuchen, etwas zu *lernen*, und Ihr *Hirn* tut sein Bestes, damit das Gelernte *nicht hängen bleibt*. Es denkt nämlich: »Wir sollten lieber ordentlich Platz für wichtigere Dinge lassen, z. B. für das Wissen darüber, welche Tiere einem gefährlich werden könnten, oder dass es eine ganz schlechte Idee ist, nackt Snowboard zu fahren.« Tja, wie schaffen wir es nun, Ihr Gehirn davon zu überzeugen, dass Ihr Leben davon abhängt, dass Sie programmieren können?

Rechnerisch denken

1 Erste Schritte

Durch die Fähigkeit, rechnerisch zu denken, haben Sie die Kontrolle. Jeder weiß, dass die Welt um einen herum immer vernetzter, konfigurierbarer, programmierbarer und eben **rechnerischer** wird. Entweder Sie bleiben passiv, oder Sie lernen zu programmieren. Sobald Sie es können, sind Sie Regisseur und Schöpfer – Sie sagen dem Computer, was er für Sie tun soll. Sobald Sie es können, bestimmen Sie Ihr Schicksal selbst (oder können zumindest Ihre Rasensprinkleranlage über das Internet steuern). Aber wie lernt man zu programmieren? Zuerst müssen Sie lernen, **rechnerisch** zu denken. Dann schnappen Sie sich eine **Programmiersprache**, um mit Ihrem Computer, Mobilgerät oder eigentlich allem mit einer CPU sprechen zu können. Und was haben Sie davon? Mehr Zeit, mehr Fähigkeiten und mehr kreative Möglichkeiten, die Dinge zu tun, die Sie wirklich tun wollen. Dann mal los …

Einfache Werte, Variablen und Typen

2 Kennen Sie Ihren Wert

Computer können nur zwei Dinge richtig gut: Werte speichern und Operationen daran ausführen. Es scheint, als täten Computer sehr viel mehr, wenn Sie Texte schicken, online einkaufen, Photoshop benutzen oder Ihr Telefon als Navi verwenden. Dennoch kann alles, was Computer tun, in **einfache Operationen** aufgeteilt werden, die an **einfachen Werten** ausgeführt werden. Deshalb besteht ein wichtiger Teil des **rechnerischen Denkens** darin, zu lernen, diese Operationen und Werte benutzen, um etwas Anspruchsvolleres, Komplexeres und Sinnvolleres zu erschaffen. Zuerst sehen wir uns aber an, um welche Werte es geht, welche Operationen Sie daran ausführen können und welche Rolle **Variablen** dabei spielen.

Boolesche Werte, Entscheidungen und Schleifen

3 Entscheidender Code

Finden Sie unsere Programme auch ein wenig langweilig?

Bis jetzt enthält unser Code einfach ein paar Anweisungen, die der Interpreter **von oben nach unten** auswertet. Keine unerwarteten Handlungswendungen, keine Überraschungen, kein unabhängiges Denken. Damit der Code interessanter wird, muss er **Entscheidungen treffen können** – um **sein eigenes Schicksal zu bestimmen** und um Dinge **mehr als einmal** ausführen zu können. Genau das lernen wir in diesem Kapitel. Danach erfahren wir mehr über ein mysteriöses Spiel namens Shoushiling und treffen einen Typ namens Boole. Außerdem sehen wir, warum ein Datentyp mit nur zwei Werten durchaus unsere Aufmerksamkeit verdient. Wir lernen sogar, wie wir die gefürchtete **Endlosschleife** bändigen können. Und los!

Listen und Iterationen

4 Etwas mehr Struktur

Zahlen, Strings und boolesche Werte sind nicht die einzigen Datentypen in Python. Bisher haben Sie beim Schreiben Ihres Codes nur sogenannte **primitive Datentypen** verwendet – Fließkomma- und Integerwerte, Strings und natürlich boolesche Werte mit Werten wie `3.14`, `42`, `„Hey, ich bin dran!“` und `True`. Und damit lässt sich schon eine Menge anfangen. Aber irgendwann müssen Sie Code schreiben, der mit großen Datenmengen umgehen muss, z. B. dem Inhalt eines Warenkorbs, den Namen aller nennenswerten Promis oder mit einem kompletten Produktkatalog. Dafür brauchen wir etwas mehr *Wumms*. In diesem Kapitel lernen wir einen neuen Datentyp kennen: die **Liste**, die eine ganze Sammlung von Werten enthalten kann. Mit Listen können Sie Daten **strukturieren**. Anstatt eine Fantastillion einzelner Variablen im Code zu verwenden, können Sie diese Variablen auch als Einheit behandeln und mithilfe der ***for***-Schleife (die wir aus dem vorigen Kapitel kennen) über die einzelnen Elemente **iterieren**. Wenn Sie dieses Kapitel abgeschlossen haben, können Sie mit Daten umgehen, die wachsen und sich ausdehnen.

Funktionen und Abstraktion

5 Funktional werden

Sie wissen bereits eine Menge. Allein mit Variablen, Datentypen, Bedingungen und Iterationen könnten Sie schon alle Programme schreiben, die Sie jemals brauchen werden. Ein Informatiker würde vermutlich sagen, dass Sie sich damit jedes nur vorstellbare Programm schreiben können. Trotzdem sind wir noch lange nicht am Ende. Im nächsten Schritt des rechnerischen Denkens lernen Sie, Ihren Code zu **abstrahieren**. Auch wenn es kompliziert klingt, wird es Ihr Leben als Programmierer sogar einfacher machen. Durch die Abstraktionen erreichen Sie einen höheren Wirkungsgrad und können komplexere und mächtigere Programme leichter erstellen. Sie können Ihren Code in praktische kleine und vor allem wiederverwendbare Einheiten verpacken. Sie brauchen sich nicht mehr um jedes kleine Detail Ihres Code zu kümmern und können stattdessen anfangen, auf einer höheren Ebene zu denken.

Sortierung und verschachtelte Iteration

4 Ordnung in die Daten bringen

Teil 2

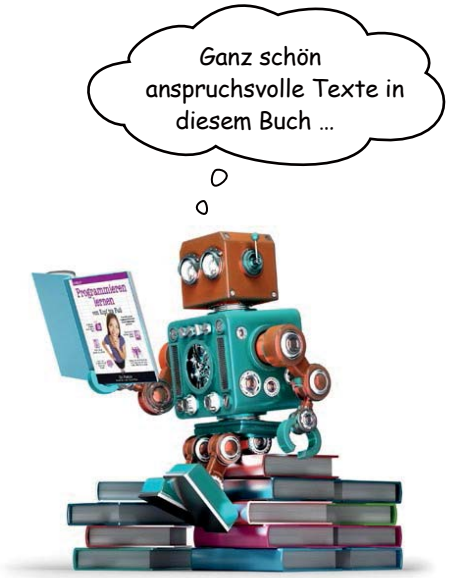
Manchmal reicht es einfach nicht aus, die Daten nur zu ordnen.

Angenommen, Sie haben eine Liste Ihrer Highscores von 80er-Jahre-Arcade-Spielen. Diese sollen alphabetisch nach Spielen sortiert werden. Und dann gibt es die Liste, in der festgehalten ist, wie oft Ihre Mitarbeiter Sie schon betrogen haben. Sie wollen wirklich gerne wissen, wer auf dieser Liste ganz oben steht. Dafür müssen Sie allerdings lernen, Daten zu sortieren. Zu dem Zweck betrachten wir ein paar Algorithmen, die es etwas mehr in sich haben als die bisher gesehenen. Wir werden uns außerdem mit verschachtelten Schleifen beschäftigen und ein wenig über die Effizienz des von Ihnen geschriebenen Codes nachdenken. Dann wollen wir Ihr rechnerisches Denken mal auf das nächste Level heben!

Text, Strings und Heuristiken

6 Die Einzelteile verbinden

Sie haben schon eine Menge Superkräfte erworben. Jetzt geht es darum, sie auch zu benutzen. In diesem Kapitel werden wir alle bisher gelernten Dinge miteinander koppeln, um damit **ziemlich coolen Code** zu schreiben. Außerdem erweitern wir Ihr Wissen und Ihre Programmierfähigkeiten. In diesem Kapitel lernen Sie, Code zu schreiben, der **sich etwas Text schnappt**, ihn in seine Einzelteile zerlegt und dann eine **Datenanalyse** daran durchführt. Außerdem erfahren Sie, was eine **Heuristik** ist. Schnallen Sie sich an, denn dies ein tiefergelegtes, hart getuntes Vollgas-Coding-Kapitel!



Module, Methoden, Klassen und Objekte

7 Die Module spielen verrückt ...

Ihr Code wird umfangreicher und komplexer. Also brauchen Sie bessere Möglichkeiten, ihn zu abstrahieren, zu modularisieren und zu organisieren. Durch Funktionen können mehrere Codezeilen zusammengefasst und wiederverwendet werden. Außerdem wissen Sie inzwischen, dass Sammlungen von Funktionen und Variablen in Modulen gespeichert und dadurch leichter weitergegeben und wiederverwendet werden können. In diesem Kapitel befassen wir uns noch einmal mit Modulen und lernen, sie effizienter zu nutzen (z. B. um den Code direkt mit anderen teilen zu können). Dann kommen wir zur ultimativen Form der Codewiederverwendung: *Objekte*. Sie werden merken, dass Sie quasi von Python-Objekten umgeben sind, die nur darauf warten, von Ihnen benutzt zu werden.

Rekursion und Dictionaries

8 Mehr als Iteration und Indizes

Es ist Zeit, Ihr rechnerisches Denken auf die nächste Ebene zu heben.

Und genau das werden wir in diesem Kapitel tun. Bisher haben wir einen eher iterativen Programmierstil verwendet. Wir haben Datenstrukturen (z. B. Listen, Strings und Zahlenbereiche) erstellt und Code geschrieben, der darüber iteriert, um zu einer Lösung zu kommen. In diesem Kapitel vermitteln wir Ihnen eine neue Weltsicht: zuerst in Bezug auf die Art, Dinge zu berechnen, und dann in Bezug auf Datenstrukturen. Rechnerisch befassen wir uns mit einer Programmiertechnik, die *rekursiven* Code verwendet – Code, der sich selbst aufruft. Wir lernen außerdem eine neue Datenstruktur kennen, die funktioniert wie ein Wörterbuch, also eher wie *assoziative Beziehungen* und weniger wie eine Liste. Danach verbinden wir die beiden neuen Konzepte und verursachen eine Menge Schwierigkeiten. Seien Sie gewarnt: Diese Themen brauchen etwas Zeit, um sich in Ihrem Gehirn festzusetzen. Aber der Aufwand lohnt sich auf jeden Fall.

Dateien speichern und auslesen

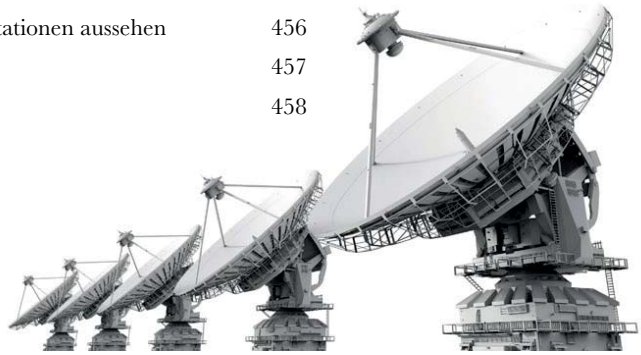
9 Persistenz

Sie wissen, dass Sie Werte in Variablen speichern können. Aber sobald Ihr Programm endet, sind sie für immer verloren. Genau da kommt die *persistente Speicherung* ins Spiel – eine Möglichkeit, Werte und Daten über die Laufzeit des Programms hinaus zu speichern. Die meisten Geräte, auf denen Python läuft, besitzen die eine oder andere Form von persistenter Speicherung, z. B. Festplatten, SD-Karten oder Cloud-Speicher. In diesem Kapitel lernen Sie, Code zu schreiben, der Daten in Dateien speichert und wieder ausliest. Und wann hilft uns das? Zum Beispiel jedes Mal, wenn Sie Benutzereinstellungen oder die Ergebnisse einer Analyse für den Chef speichern wollen, ein Bild zur Bearbeitung einlesen, die E-Mails der letzten zehn Jahre durchsuchen oder bestimmte Daten in einer Tabellenkalkulation berechnen wollen. Wir könnten die Liste noch endlos weiterführen, aber wir fangen wohl besser mit dem Kapitel an.

Web-APIs benutzen

10 Sie sollten wirklich öfter mal ausgehen

Sie haben großartigen Code geschrieben, aber Sie sollten wirklich öfter ausgehen. Im Web wartet die ganze Welt der **Daten** auf Sie: Brauchen Sie Informationen zum Wetter? Oder wollen Sie auf eine riesige Datenbank mit Kochrezepten zugreifen? Oder stehen Sie mehr auf Sportergebnisse? Vielleicht eine Musikdatenbank mit Künstlern, Alben und Titeln? Mit **Web-APIs** brauchen Sie nur zuzugreifen. Um APIs zu benutzen, benötigen Sie lediglich ein paar Informationen über ihre Funktionsweise, die Aussprache des lokalen Dialekts und die Verwendung einiger neuer Python-Module: `requests` und `json`. In diesem Kapitel werden wir Web-APIs erforschen und Ihre Python-Fähigkeiten auf eine neue Ebene heben. Tatsächlich nehmen wir Sie mit in den Weltraum – und wieder zurück.



Widgets, Events und emergentes Verhalten

11 Interaktivität

Sie haben inzwischen schon einige grafische Applikationen geschrieben. Eine echte grafische Benutzerschnittstelle fehlt aber noch. Das heißt, Sie haben noch nichts programmiert, mit dessen Hilfe der Benutzer mit einer grafischen Benutzeroberfläche (auch als GUI bezeichnet) interagieren kann. Hierfür brauchen Sie eine neue Denkweise dazu, wie ein Programm ausgeführt wird, sodass es direkt auf die Benutzereingaben **reagiert**. Hat der Benutzer gerade einen Button angeklickt? Dann sollte Ihr Code auch wissen, wie er darauf reagieren muss und was als Nächstes zu tun ist. Die Programmierung von Benutzeroberflächen unterscheidet sich stark von der bisher verwendeten prozeduralen Vorgehensweise. In diesem Kapitel werden Sie Ihr erstes echtes GUI programmieren. Das wird übrigens weder ein einfacher To-do-Listenmanager noch ein Höhe/Breite-Rechner. Wir machen etwas viel Interessanteres. Wir schreiben einen Simulator für künstliches Leben mit emergentem (sich weiterentwickelndem) Verhalten. Und was heißt das? Finden Sie es heraus.

Objektorientierte Programmierung

12 Ausflug nach Objekthausen

In diesem Buch haben Sie meist Funktionen benutzt, um Code zu abstrahieren. Bisher haben Sie für die Programmierung einen **prozeduralen Ansatz** verwendet: einfache Anweisungen, Bedingungen und `for`/`while`-Schleifen mit Funktionen. Nichts davon ist wirklich **objektorientiert**. Eigentlich ist es *überhaupt nicht* objektorientiert! Im Code haben wir uns Objekte und ihre Verwendung bereits angesehen. Aber eigene Objekte haben Sie bisher nicht erstellt, und mit dem objektorientierten Design Ihres Code haben Sie sich auch noch nicht beschäftigt. Daher ist es jetzt Zeit, diese langweilige prozedurale Stadt hinter sich zu lassen. In diesem Kapitel finden Sie heraus, warum die Verwendung von Objekten Ihr Leben deutlich erleichtern kann – zumindest bei der **Programmierung** (für Unterstützung in anderen Lebensbereichen reicht dieses Buch einfach nicht aus). Nur eine Warnung: Sobald Sie Objekte entdeckt haben, wollen Sie nicht mehr zurück. Schicken Sie uns eine Postkarte, wenn Sie angekommen sind.