

## Inhalt (im Überblick)

	Einführung	xxi
1	Erste Schritte: <i>Rechnerisch denken</i>	1
2	Kennen Sie Ihren Wert: <i>Einfache Werte, Variablen und Typen</i>	33
3	Entscheidender Code: <i>Boolesche Werte, Entscheidungen und Schleifen</i>	73
4	Etwas mehr Struktur: <i>Listen und Iterationen</i>	125
5	Funktional werden: <i>Funktionen und Abstraktion</i>	179
4b	Ordnung in die Daten bringen: <i>Sortierung und verschachtelte Iteration</i>	225
6	Die Einzelteile verbinden: <i>Text, Strings und Heuristiken</i>	245
7	Die Module spielen verrückt: <i>Module, Methoden, Klassen und Objekte</i>	291
8	Mehr als Iteration und Indizes: <i>Rekursion und Dictionaries</i>	341
9	Persistenz: <i>Dateien speichern und auslesen</i>	393
10	Sie sollten wirklich öfter mal ausgehen: <i>Web-APIs benutzen</i>	435
11	Interaktivität: <i>Widgets, Events und emergentes Verhalten</i>	467
12	Ausflug nach Objekthausen: <i>Objektorientierte Programmierung</i>	523
	Anhang: Die Top Ten der Themen, die wir nicht behandelt haben: <i>Was übrig bleibt</i>	575
	Index	587

## 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?



Einführung	xxi
Für wen ist dieses Buch?	xxii
Wir wissen, was Sie gerade denken.	xxiii
Lies mich!	xxviii
Sie müssen Python installieren	xxx
Ein paar Worte zur Organisation Ihres Codes	xxxii
Danksagungen	xxxiii
Das Gutachterteam	xxxiv

# 1

Rechnerisch denken

## 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 ...



Immer der Reihe nach	2
Wie Programmieren funktioniert	6
Sprechen wir überhaupt die gleiche Sprache?	7
Die Welt der Programmiersprachen	8
Code mit Python schreiben und ausführen	13
Eine kurze Geschichte von Python	15
Python auf Herz und Nieren prüfen	18
Ihre Arbeit speichern	20
Herzlichen Glückwunsch zu Ihrem ersten Python-Programm!	21
Phrasendrescher	25
Den Code in die Maschine bekommen	26

## 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.

Einen Rechner für Hundejahre programmieren	34
Pseudocode in richtigen Code umwandeln	36
Schritt 1: Eingaben einsammeln	37
Wie die input-Funktion funktioniert	38
Variablen verwenden, um Werte zu erinnern und zu speichern	38
Benutzereingabe einer Variablen zuweisen	39
Schritt 2: Weitere Eingaben einsammeln	39
Zeit, unseren Code auszuführen	40
Etwas Code eingeben	43
Ein genauer Blick auf Variablen	44
Etwas mehr Ausdruck	45
Variablen sind VARIABLE	46
Besser leben durch Operatoren-Präzedenz	47
Rechnen mit Operatoren-Präzedenz	48
Finger von der Tastatur!	51
Schritt 3: Das Alter des Hundes berechnen	52
Houston, wir haben ein Problem!	53
Fehler <del>sind menschlich</del> gehören zum Programmieren dazu	54
Etwas mehr Debugging ...	56
Welcher Python-(Daten-)Typ sind Sie?	58
Den Code reparieren	59
Houston, wir sind abgehoben!	60
Schritt 4: Benutzerfreundliche Ausgaben	61



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!

Wollen Sie ein Spiel spielen?	74
Zuerst ein allgemeines Konzept	76
Die Wahl des Computers	77
Wie Zufallszahlen genutzt werden	78
Vorstellung des booleschen Datentyps	81
Entscheidungen treffen	82
Entscheidungen und noch mehr Entscheidungen	83
Zurück zu Stein, Schere, Papier	84
Die Wahl des Benutzers ermitteln	85
Ein Blick auf die Wahl des Benutzers	88
Den Code zum Finden eines Unentschiedens einbauen	89
Wer hat gewonnen?	90
Implementieren wir die Spiellogik	92
Mehr über boolesche Operatoren	93
Ist das dokumentiert?	98
Den Code mit Kommentaren versehen	99
Wir müssen das Spiel endlich fertigstellen!	100
Woher wissen wir, ob eine Benutzereingabe ungültig ist?	101
Den Ausdruck überprüfen und aufräumen	102
Den Benutzer immer wieder zur Eingabe auffordern	104
Dinge mehr als einmal tun	105
Wie die while-Schleife funktioniert	106
while benutzen, um den Benutzer zur Eingabe aufzufordern, bis eine gültige Auswahl getroffen wurde	110
Glückwunsch! Sie haben Ihr erstes Spiel programmiert!	112



## 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.

Können Sie Bubbles-R-Us helfen?	126
Mehrere Werte in Python darstellen	127
Wie Listen funktionieren	128
Wie lang ist die Liste eigentlich?	131
Auf das letzte Listenelement zugreifen	132
Python macht das sogar noch einfacher	132
Pythons negative Indizes benutzen	133
In der Zwischenzeit bei Bubbles-R-Us ...	135
Wie man über eine Liste iteriert	138
Den Ausgabefehler beseitigen	139
Den Ausgabefehler wirklich beseitigen	140
Die for-Schleife – der bevorzugte Weg zum Iterieren über eine Liste	142
Die for-Schleife mit einer Zahlenfolge benutzen	145
Mehr tun mit Zahlenfolgen	146
Die Einzelteile zusammensetzen	148
Eine komplett neue Liste erstellen	156
Mit Listen geht noch mehr	157
Probefahrt für den fertigen Bericht	161
Und die Gewinner sind ...	161
Probefahrt für die kosteneffektivste Seifenblasenmischung	165



# 5 Funktionen und Abstraktion 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.

Was stimmt denn mit dem Code nicht?	181
Einen Codeblock in eine FUNKTION umwandeln	183
Wir haben eine Funktion erstellt, aber wie können wir sie benutzen?	184
Aber wie funktioniert das wirklich?	184
Funktionen können auch Dinge ZURÜCKGEBEN	192
Funktionen mit Rückgabewerten aufrufen	193
Ein bisschen Refaktorisierung zwischendurch	195
Den Code ausführen	196
Den Avatar-Code abstrahieren	197
Den Körper der get_attribute-Funktion schreiben	198
get_attribute aufrufen	199
Wir sollten noch etwas mehr über Variablen sprechen ...	201
Den Geltungsbereich von Variablen verstehen	202
Wenn Variablen an Funktionen übergeben werden ...	203
Die drink_me-Funktion aufrufen	204
Was ist mit der Verwendung globaler Variablen in Funktionen?	207
Mehr zu Parametern: Standardwerte und Schlüsselwörter	210
Wie Standardwerte funktionieren	210
Geben Sie erforderliche Parameter immer zuerst an!	211
Argumente mit Schlüsselwörtern verwenden	212
Wie man diese Optionen einordnen sollte	212



Sortierung und verschachtelte Iteration

4  
Teil 2*Ordnung in die Daten bringen***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!

Bubble Sort verstehen	228
Wir beginnen mit Durchgang 1	228
Etwas Bubble Sort-Pseudocode	231
Bubble Sort in Python implementieren	234
Die Nummern der Seifenblasenmischungen berechnen	236



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 getunttes Vollgas-Coding-Kapitel!

Willkommen zu den Datenwissenschaften	246
Wie berechnet man einen Lesbarkeitsindex?	247
Der große Plan	248
Ein bisschen Pseudocode	249
Wir brauchen etwas Text zur Analyse	250
Die Funktion erstellen	252
Als Erstes brauchen wir die Anzahl der Wörter in unserem Text	253
Die Anzahl der Sätze berechnen	257
Die count_sentences-Funktion schreiben	258
Die Anzahl der Silben zählen – oder: Heuristiken lieben lernen	264
Die Heuristik vorbereiten	267
Die Heuristik schreiben	268
Wie man Vokale zählt	269
Aufeinanderfolgende Vokale ignorieren	269
Den Code zum Ignorieren aufeinanderfolgender Vokale schreiben	270
Die Buchstaben e und y sowie Interpunktionszeichen am Satzende entfernen	272
Slicing (und Substrings) benutzen	274
Den heuristischen Code fertigstellen	276
Die Formel für den Lesbarkeitsindex implementieren	278
Und noch mehr	283



## 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.

Ein schneller Rückblick auf Module	294
Die globale Variable <code>__name__</code>	296
<code>analyze.py</code> aktualisieren	297
<code>analyze.py</code> als Modul benutzen	299
<code>analyze.py</code> mit Docstrings versehen	301
Andere Python-Module erforschen	305
Moment mal. Haben Sie gerade »Schildkröten« gesagt?!	306
Erschaffen Sie Ihre eigene Schildkröte!	308
Tierversuche mit Schildkröten	309
Eine zweite Schildkröte hinzufügen	311
Was sind diese Schildkröten überhaupt?	314
Was sind Objekte?	315
Okay, und was ist eine Klasse?	316
Eine Klasse ist kein Objekt. Sie wird verwendet, um Objekte zu erzeugen.	316
Objekte und Klassen verwenden	318
Worum geht es bei den Methoden und Attributen?	319
Klassen und Objekte überall	320
Bereitmachen zum Schildkrötenrennen	322
Das Spiel planen	323
Fangen wir mit dem Programmieren an	324
Das Spiel vorbereiten	324
Den Setup-Code schreiben	325
Nicht so schnell!	326
Das Rennen beginnen	328

Gute Arbeit! Dank Ihrer großartigen Dokumentation habe ich schnell verstanden, wie das `analyze`-Modul funktioniert.



CRIME SCENE DO NOT ENTER

CRIME SCENE DO NOT ENTER

CRIME SCENE DO NOT ENTER

## 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.



Eine andere Art, Dinge zu berechnen	342
Kommen wir jetzt zu etwas völlig anderem ...	343
Schreiben wir etwas Code für beide Fälle	344
Ein wenig mehr Übung	347
Rekursion zum Finden von Palindromen benutzen	348
Einen rekursiven Palindrom-Detektor schreiben	349
Das Unsoziale Netzwerk	360
Das Dictionary	362
Sehen wir, wie ein Dictionary erstellt wird.	362
Schlüssel und Werte müssen keine Strings sein.	363
Natürlich können Sie die Schlüssel auch wieder entfernen.	363
Vielleicht wollen Sie aber zuerst überprüfen, ob er überhaupt existiert.	363
Kann man über Dictionaries auch iterieren?	364
Dictionaries für das Unsoziale Netzwerk einsetzen	366
Aber wie fügen wir weitere Attribute hinzu?	368
Erinnern Sie sich an das Killer-Feature unseres Unsozialen Netzwerks?	370
Den unsozialsten Benutzer finden	371
Können wir uns die Ergebnisse der Funktionsaufrufe nicht einfach merken?	376
Ein Dictionary zum Speichern der Fibonacci-Ergebnisse	376
Etwas gegen teure Berechnungen: Memoisation	377
Ein genauer Blick auf die koch-Funktion	380
Das Koch-Fraktal eingehend erforschen	382

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.

Bereit für Crazy Libs?	394
Wie Crazy Libs funktionieren soll	396
Schritt 1: Text der Geschichte aus einer Datei lesen	399
Dateipfade richtig benutzen	400
Relative Pfade	400
Absolute Pfade	401
Oh, und vergessen Sie nicht, wieder aufzuräumen, wenn Sie fertig sind.	402
Dateien im Python-Code lesen	403
Jetzt hör schon auf!	406
Verwendung der <code>readline</code> -Methode am Dateiojekt	407
Woher wissen wir, wann die letzte Zeile gelesen wurde?	409
Eine Crazy Lib-Schablone einlesen	410
Den Schablonentext verarbeiten	411
Den Bug mit einer neuen Stringmethode beseitigen	413
Den Bug tatsächlich beheben	414
Der eine oder andere Code hat echte Probleme	415
Mit Ausnahmen umgehen	417
Ausdrücklich mit Ausnahmen umgehen	418
Crazy Libs aktualisieren, um mit Ausnahmen umzugehen	420
Unser letzter Schritt: Das Crazy Lib speichern	421
Den Rest des Code aktualisieren	421

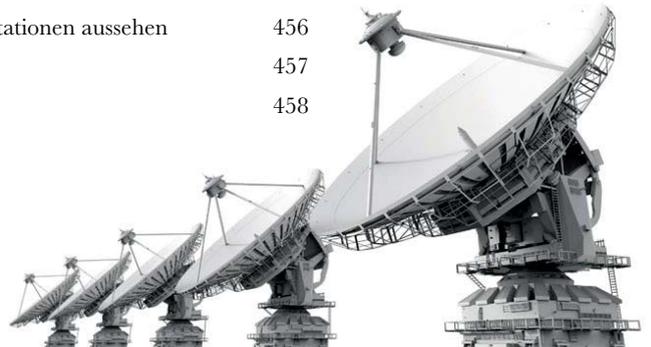


Web-APIs benutzen

# 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.

Erweitern Sie Ihre Reichweite mit Web-APIs	436
Die Funktionsweise von Web-APIs	437
Alle Web-APIs besitzen eine Webadresse	438
Zeit für ein schnelles Upgrade	441
Das Upgrade durchführen	442
Jetzt brauchen wir nur noch eine gute Web-API ...	443
Ein genauerer Blick auf die API	444
Web-APIs verwenden JSON für die Rückgabe der Daten	445
Sehen wir uns das <code>requests</code> -Modul noch einmal an	447
Die Einzelteile zusammensetzen: einen Request an Open Notify verschicken	449
JSON in Python verarbeiten	450
Das <code>JSON</code> -Modul zum Auslesen der ISS-Daten benutzen	451
Wir brauchen etwas mehr Grafik	452
Das <code>screen</code> -Objekt	453
Wir benutzen eine Schildkröte, um die ISS darzustellen	455
Schildkröten können auch wie Weltraumstationen aussehen	456
Vergessen Sie die ISS. Wo sind WIR?	457
Den ISS-Code fertigstellen	458



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.

Willkommen in der WUNDERVOLLEN WELT des KÜNSTLICHEN LEBENS	468
Ein genauerer Blick auf das Spiel des Lebens	469
Was wir bauen werden	472
Haben wir das richtige Design?	473
Das Datenmodell (Model) entwickeln	477
Das Raster im Code abbilden	477
Eine Generation des Spiels des Lebens berechnen	478
Das Schicksal aller Zellen berechnen	478
Den Model-Code fertigstellen	482
Den View erstellen	485
Das erste Widget erstellen	486
Die fehlenden Widgets hinzufügen	487
Das Layout korrigieren	488
Die Widgets auf einem Gitter anordnen	489
Das Gitter-Layout in Code übersetzen	489
Mit dem Controller weitermachen	491
Eine Aktualisierungsfunktion einbauen	491
Bereit für eine neue Art der Programmierung?	494
Die Funktionsweise des Start/Pause-Buttons	497
Eine andere Art von Event	499
Die nötige Technologie haben wir: die after-Methode	501
Zellen direkt eingeben und bearbeiten	504
Den grid_view-Event-Handler schreiben	505
Jetzt können wir ein paar Muster hinzufügen.	506
Einen Handler für das Auswahlmü schreiben	507
Die Ladefunktion für das Muster schreiben	510
Noch weiter!	517



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.

Die Dinge anders aufteilen	524
Worum geht es bei der objektorientierten Programmierung überhaupt?	525
Ihre erste Klasse planen	527
Ihre erste Klasse schreiben	528
Wie der Konstruktor funktioniert	528
Die bark-Methode schreiben	531
Die Funktionsweise von Methoden	532
Vererbung nutzen	534
Die ServiceDog-Klasse implementieren	535
Ein genauerer Blick auf abgeleitete Klassen	536
Ein ServiceDog IST EIN Dog	537
IST-Beziehungen im Code überprüfen	538
Verhalten überschreiben und erweitern	542
Noch mehr Fachjargon ...	544
Objekt HAT ein anderes Objekt	546
Ein Hundehotel entwickeln	549
Das Hundehotel implementieren	550
Das Hundehotel renovieren	553
Das Hotel um ein paar Aktivitäten erweitern	554
Ich kann alles, was du auch kannst, oder: Polymorphismus	555
Es ist Zeit, den Hunden das Gassigehen beizubringen	556
Die Macht der Vererbung	558

