Ein Buch zum Mitmachen und Verstehen

Programmieren

von Kopf bis Fuß

Eine allgemeine Einführung, deren Beispiele in Python geschrieben sind.

Laden Sie sich die wichtigen Konzepte direkt in Ihr Hirn

Bestücken Sie Ihren Werkzeugkasten mit Methoden, Funktionen und Objekten

Vermeiden Sie nervige Ein-/Ausgabe-Pannen

Verarbeiten Sie Ihre Daten wie ein echter Profi

Bauen Sie so funktionelle wie attraktive grafische Anwendungen

Erfahren Sie, wie sich wiederkehrende Aufgaben automatisieren lassen

O'REILLY®

Paul Barry & David Griffiths
Deutsche Übersetzung von Lars Schulten

David Griffiths / Paul Barry, Programmieren von Kopf bis Fuß, O´Reilly, ISBN 9783897219922

Der Inhalt (im Überblick)

Der 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, 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, etwas über das Programmieren zu wissen?

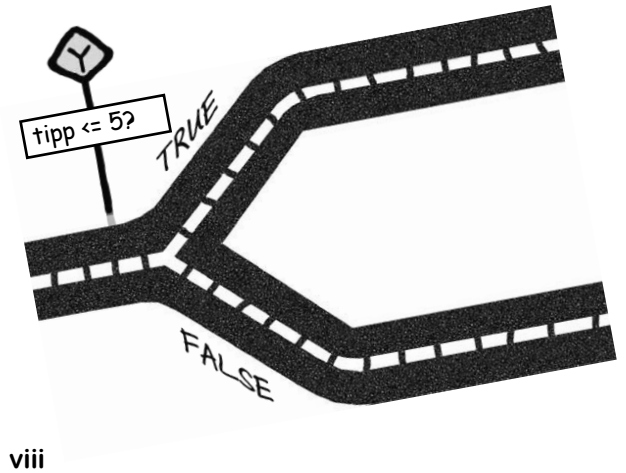
Der Anfang des Programmierens

1 Orientierungshilfe

Eigene Programme verleihen Ihnen die Macht, Ihren PC zu steuern.

Fast jeder weiß, wie man mit einem Computer *arbeitet*, aber nur wenige machen den nächsten Schritt und lernen, wie man ihn *steuert*. Wenn Sie Software nutzen, die von anderen entwickelt wurde, sind Sie immer auf das beschränkt, was Sie *deren* Meinung nach tun möchten. Schreiben Sie Ihre eigenen Programme, und nur noch Ihre eigene Vorstellungskraft kann Sie einschränken. Das Programmieren macht Sie kreativer, lässt Sie präziser denken und lehrt Sie, Probleme logisch zu analysieren und zu lösen.

Wollen Sie lieber programmiert oder Programmierer sein?

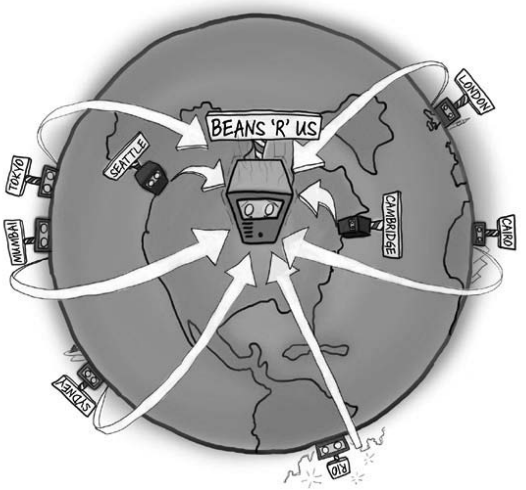


Textdaten

2 Jeder Text zu seiner Zeit

Stellen Sie sich vor, Sie müssten ohne Worte kommunizieren.

Alle Programme verarbeiten Daten – und eine der wichtigsten Arten von Daten ist **Text**. In diesem Kapitel werden Sie das Basiswissen zu **Textdaten** erhalten. Sie werden Text **suchen** und automatisch genau das bekommen, **was Sie suchten**. Und unterwegs werden Sie grundlegende Konzepte der Programmierung wie **Methoden** aufsammeln und erfahren, wie Sie sie einsetzen können, **um sich Ihre Daten gefügig zu machen**. Und schließlich werden Sie Ihren Programmen mithilfe von **Codebibliotheken** im Handumdrehen **Superkräfte geben**.



Funktionen

3 Ordnung schaffen

Wenn Programme wachsen, wird Code unübersichtlich.

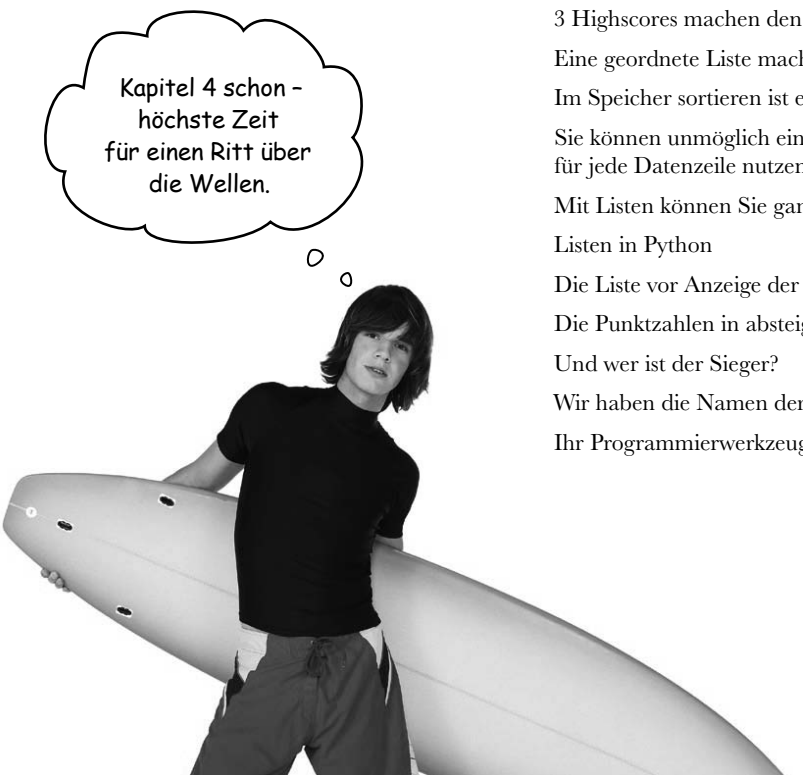
Unübersichtlicher und komplexer Code kann schwer zu lesen und noch schwerer zu warten sein. Eine Möglichkeit, diese Komplexität in den Griff zu bekommen, ist der Einsatz von **Funktionen**. Funktionen sind **Codeeinheiten**, die Sie nach Bedarf in Ihren Programmen einsetzen. Sie ermöglichen Ihnen, **gemeinsam genutzte Aktionen auszulagern**, und das bedeutet, dass Sie Ihren Code **besser lesbar** und **leichter wartbar** machen. In diesem Kapitel werden Sie entdecken, wie Ihnen etwas Wissen zu Funktionen **das Programmiererleben erheblich vereinfachen kann**.

Daten in Dateien und Listen

4 Sortieren

Mit Ihren Programmen entwickeln sich auch die Datenverarbeitungsanforderungen.

Und wenn Sie mit vielen Daten arbeiten müssen, wird es schnell recht mühselig, einzelne Variablen für jedes einzelne Datenelement zu verwenden. Programmierer nutzen deswegen ziemlich beeindruckende Behälter oder Container (die als **Datenstrukturen** bezeichnet werden), um sich die Arbeit mit vielen Daten zu vereinfachen. Häufig kommen alle Daten aus Dateien, die auf der Festplatte gespeichert sind. Aber wie arbeitet man mit Daten in Dateien? Es wird sich herausstellen, dass es ein Kinderspiel ist. Blättern Sie um und erfahren Sie, warum!



Abbildungen und Datenbanken

5 Daten ihren Platz zuweisen

Listen sind nicht der einzige Datencontainer.

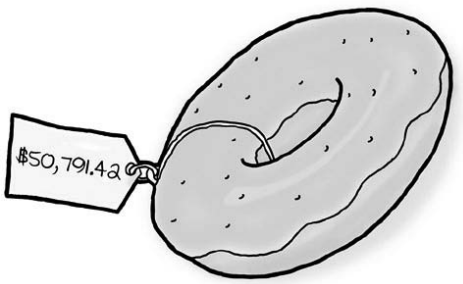
Programmiersprachen bieten weitere Spielzeuge zum Festhalten von Daten, und unser erwähltes Werkzeug, Python, macht da keine Ausnahme. In diesem Kapitel werden Sie Werte mit Namen **verknüpfen** und dazu eine Datenstruktur nutzen, die unter vielen Namen bekannt ist, Hash, Tabelle, **Abbildung**, und unter Python *Dictionary*. Und bei der Speicherung von Daten wollen wir uns jetzt nicht mehr auf Textdaten in Dateien beschränken, sondern auf Daten in einem *externen Datenbanksystem zugreifen*. **Information** ist alles, und da es die ohne Daten nicht gibt, blättern Sie um und beginnen damit, Ihre stetig wachsenden Programmierfertigkeiten auf einige coole Datenverarbeitungsaufgaben anzuwenden.

Modulare Programmierung

6 In der Spur bleiben

Ihr Code geht in viele Programme ein.

Und obgleich **Teilen** eine gute Sache ist, müssen Sie *aufpassen*. Es könnte sein, dass Programmierer Ihren Code nehmen und auf **unerwartete** Weise nutzen oder andere ihn einfach ändern, ohne Sie darüber zu informieren. Eventuell möchten Sie eine Funktion in all Ihren Programmen nutzen, und mit der Zeit **ändert** sich der Code dieser Funktion, damit er weiterhin Ihre Anforderungen erfüllt. Schlaue Programmierer nutzen *modulare Programmiertechniken*, um ihre Arbeitsbelastung unter Kontrolle zu halten. Finden Sie auf den folgenden Seiten heraus, wie man das anstellt ...



Grafische Benutzeroberflächen

7 Augen- und Ohrenschmaus

Ihre Fertigkeiten als Programmierer nehmen stetig zu.

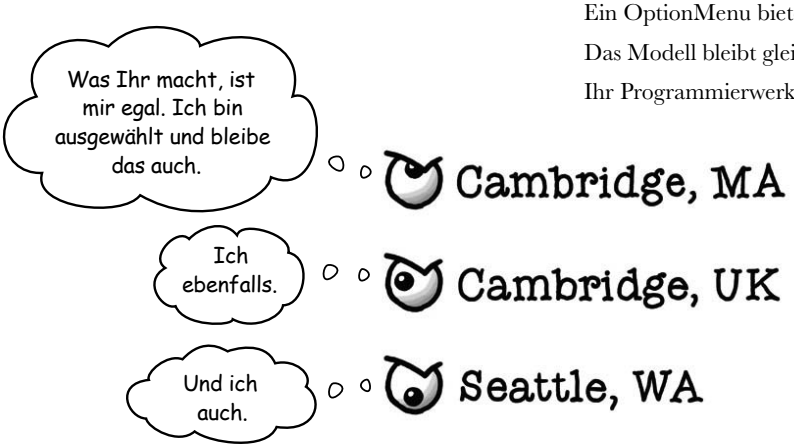
Aber es ist ein Jammer, dass Ihre Programme nicht *ansehnlicher* sind. Fragen und Antworten auf der Konsole anzeigen zu können, ist ja ganz hübsch, aber doch recht retro, oder nicht? Fehlt eigentlich nur noch grüner Text auf schwarzem Hintergrund zum wahren IT-Steinzeitgefühl. Es muss doch eine *bessere* Möglichkeiten geben, mit Benutzern zu kommunizieren – und die gibt es natürlich: **grafische Benutzeroberflächen** oder **GUIs**. Klingt cool und komplex und kann beides sein. Aber keine Angst, mit ein zwei Tricks haben Sie Ihren grafischen Code in null Komma nichts laufen.

GUIs und Daten

8 Grafische Dateneingabe

GUIs verarbeiten nicht nur Ereignisse, sondern auch Daten.

Fast alle GUI-Anwendungen müssen Benutzerdaten lesen, und die Wahl der richtigen Eingabeelemente kann entscheiden, ob Ihre Benutzeroberfläche eine *Dateneingabehölle* oder der siebte *Benutzerhimmel* ist. Widgets können einfachen Text akzeptieren oder nur eine Optionsliste präsentieren. Es gibt viele verschiedene Widgets, und das heißt, dass Sie viele Optionen haben. Welche Sie davon wählen, kann natürlich ganz entscheidende Auswirkungen haben. Es ist an der Zeit, dass Sie Ihre GUI-Programme **weiterentwickeln**.



8½

Ausnahmen und Dialogfenster

Nachricht angekommen?

Manchmal geht etwas schief, und Sie müssen sich darum kümmern.

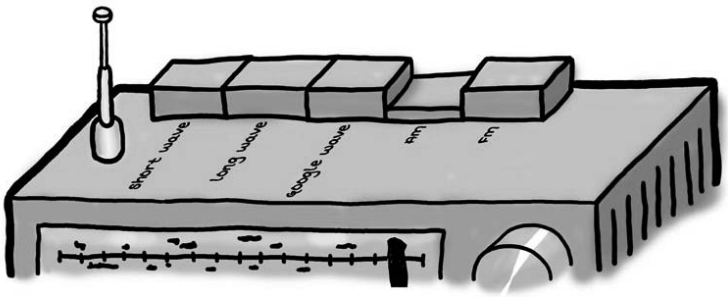
Es gibt immer Dinge, die Sie nicht im Griff haben. Netzwerke fallen aus. Dateien verschwinden. Schlaue Programmierer wissen, wie sie mit derartigen **Fehlern** umgehen, und bieten in ihren Programmen **Ausweichlösungen**. Gute Software informiert den Nutzer, wenn etwas Übles passiert, und sagt ihm, was erforderlich ist, um die Angelegenheit zu regeln. Wenn Sie lernen, wie Sie **Ausnahmen** und **Dialogfenster** einsetzen, können Sie die Zuverlässigkeit und Qualität Ihrer Software steigern.

Elemente grafischer Oberflächen

9 Das richtige Werkzeug wählen

Gut verwendbare Programme zu schreiben, ist nicht schwer..

Bei GUI-Anwendungen gibt es einen ganz entscheidenden Unterschied zwischen *funktionierenden* Schnittstellen und **nützlichen** sowie **effektiven** Schnittstellen. Wie man das richtige Werkzeug für eine Aufgabe wählt, lernt man durch Erfahrung, und die erwirbt man nur, indem man mit den verfügbaren Werkzeugen arbeitet. In diesem Kapitel werden Sie Ihre Fertigkeiten im Aufbau von GUI-Anwendungen erweitern. Es gibt noch eine Menge nützlicher Widgets, die auf Sie warten. Blättern Sie also um und lassen Sie uns fortfahren.

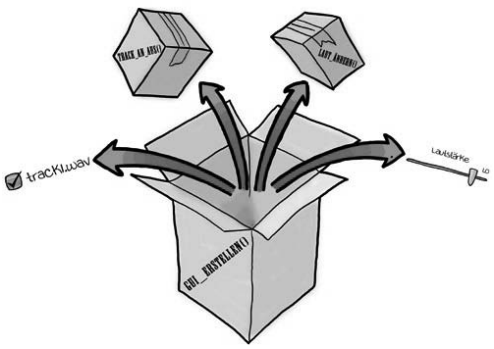


Eigene Widgets und Klassen

10 Objekte im Sinn

Anforderungen können komplex sein, Programme müssen es nicht sein.

Indem Sie Objektorientierung nutzen, können Sie Ihre Programme sehr **leistungsfähig** machen, ohne dazu Unmengen an Code zusätzlich schreiben zu müssen. Lesen Sie weiter und erfahren Sie, wie Sie **eigene Widgets** erstellen, die genau das machen, was Sie *wollen*, und Ihnen dazu verhelfen, **Ihre Programmierfertigkeiten weiterzuentwickeln**.

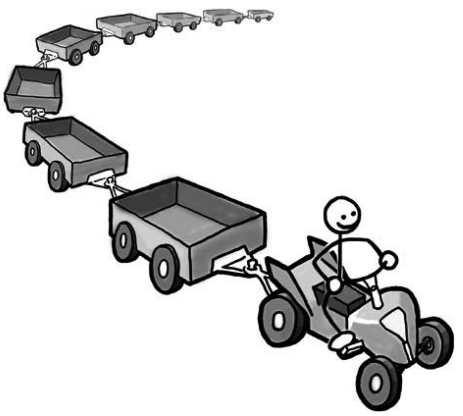


Anhang: Was übrig bleibt

Die Top-Ten der Dinge (die wir nicht behandelt haben)

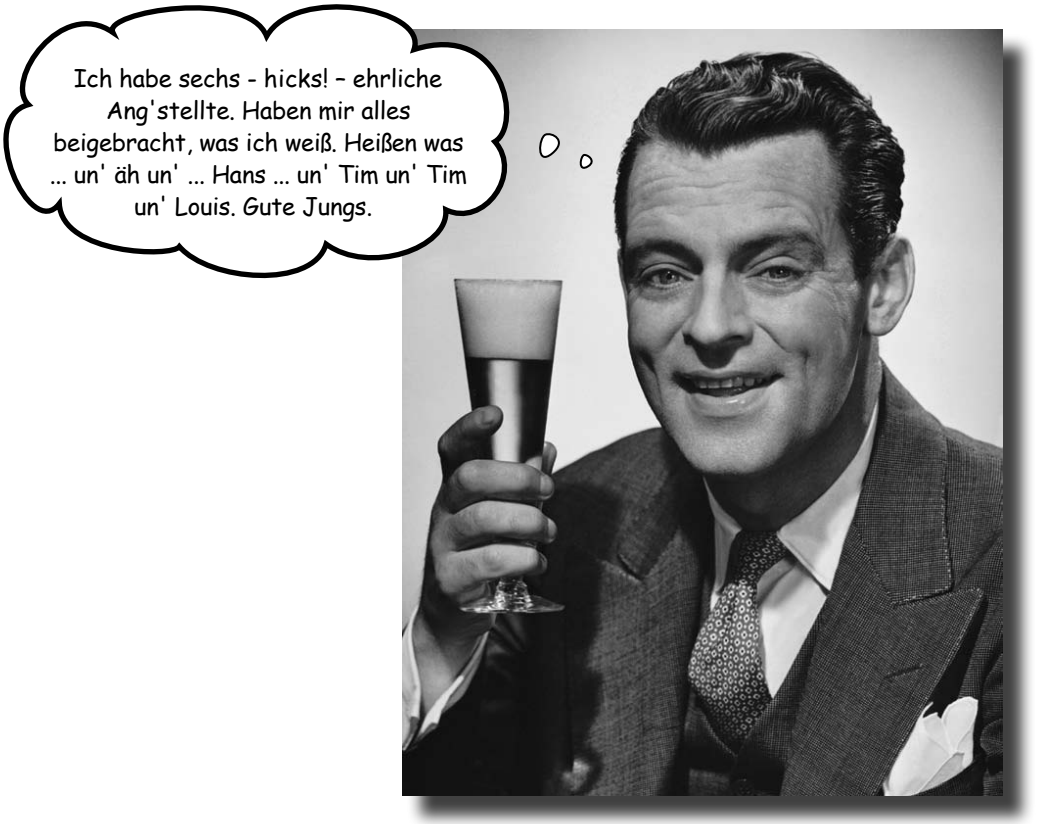
Sie haben einen weiten Weg zurückgelegt.

Doch programmieren zu lernen ist eine Aufgabe, die nie abgeschlossen ist. Je mehr Sie programmieren, umso wichtiger wird es, dass Sie **neue Wege erforschen, um bestimmte Dinge zu tun**. Sie müssen sich mit **neuen Werkzeugen** und **neuen Techniken** vertraut machen. Leider bietet dieses Buch einfach nicht genügend Raum dafür, Ihnen all das zu zeigen, was eventuell nützlich für Sie werden könnte. Deswegen finden Sie hier eine Liste der zehn wichtigsten Dinge, die wir nicht behandelt haben, die Sie vielleicht aber als Nächstes lernen sollten.



2 Textdaten

Jeder Text zu seiner Zeit



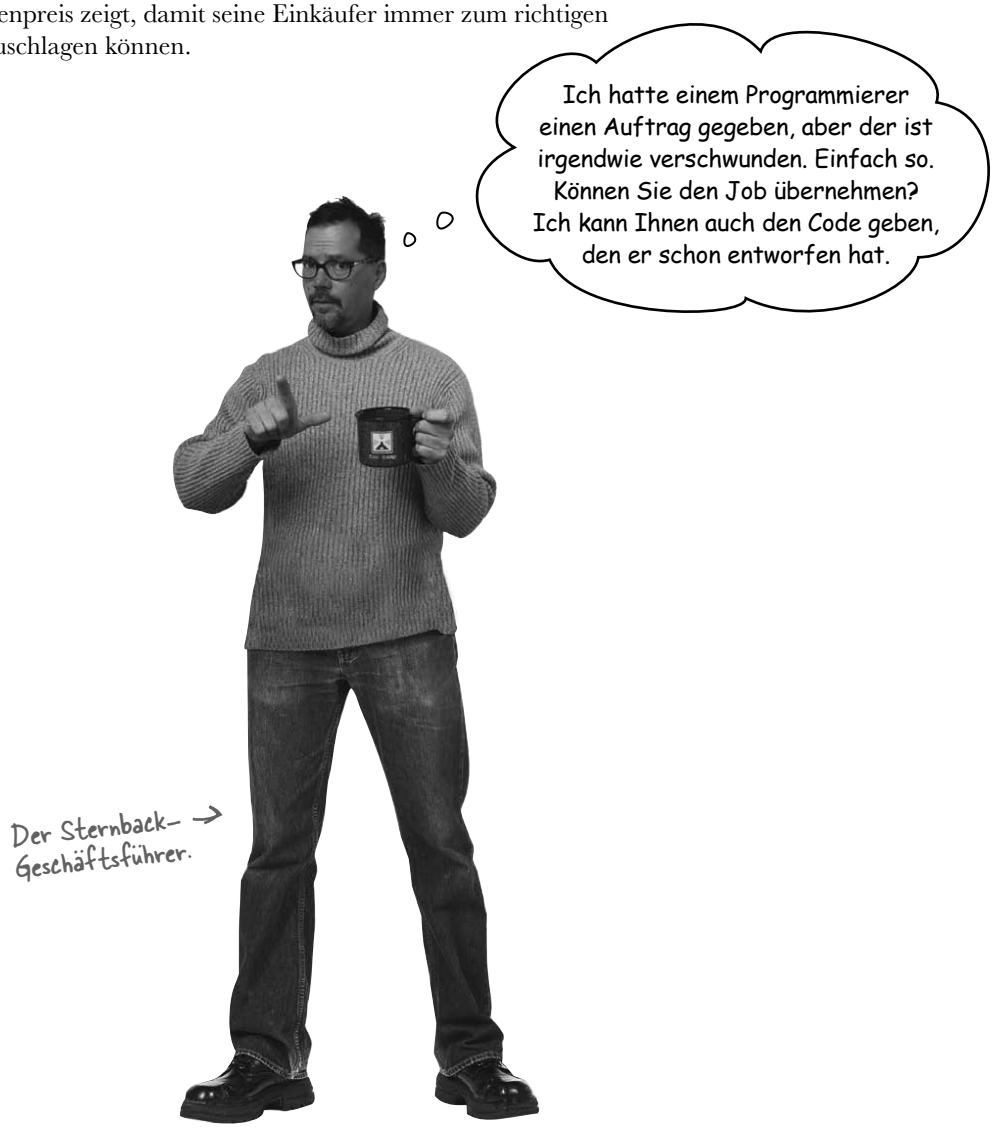
Stellen Sie sich vor, Sie müssten ohne Worte kommunizieren.

Alle Programme verarbeiten Daten – und eine der wichtigsten Arten von Daten ist **Text**. In diesem Kapitel werden Sie das Basiswissen zu **Textdaten** erhalten. Sie werden Text **suchen** und automatisch genau das bekommen, **was Sie suchten**. Und unterwegs werden Sie grundlegende Konzepte der Programmierung wie **Methoden** aufsammeln und erfahren, wie Sie sie einsetzen können, **um sich Ihre Daten gefügig zu machen**. Und schließlich werden Sie Ihren Programmen mithilfe von **Codebibliotheken** im Handumdrehen **Superkräfte geben**.

Der neue Sternback-Auftrag

Sternback-Kaffee ist als die am schnellsten wachsende Kaffeehauskette bekannt. Stehen Sie vor der Filiale bei Ihnen um die Ecke, müssen Sie nur über die Straße schauen und sehen mit Sicherheit gleich die nächste.

Der Geschäftsführer von Sternback ist permanent auf der Suche nach Möglichkeiten, den Profit zu steigern, und hatte mal wieder eine umwerfende Idee. Er möchte ein Programm, das ihm den aktuellen Kaffeebohnenpreis zeigt, damit seine Einkäufer immer zum richtigen Zeitpunkt zuschlagen können.



Hier ist der aktuelle Sternback-Code

Der andere Programmierer hatte schon mit dem Programmieren begonnen und wir können seinen Code als Grundlage nutzen. So sieht der Python-Code aus, den Ihnen der Sternback-Boss zugeschickt hat. Die Frage ist nur, was dieser Code eigentlich macht.

Hier ist der Programmcode in seiner aktuellen Form.

```
import urllib.request

seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices.html")
text = seite.read().decode("utf8")

print(text)
```


KOPF-NUSS

Schauen Sie sich den aktuellen Sternback-Code genau an. Was, meinen Sie, macht er?

PROBEFAHRT

Geben Sie den Code in IDLE ein, speichern Sie das Programm und führen Sie es aus.

Hier ist der in IDLE eingegebene Programmcode.

html.py – /Code/Kapitel2/html.py

```
import urllib.request

seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices.html")
text = seite.read().decode("utf8")

print(text)
```

Python Shell

```
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
<html><head><title>Welcome to the Beans'R'Us Pricing Page</title>
<link rel="stylesheet" type="text/css" href="beansrus.css" />
</head><body>
<h2>Welcome to the Beans'R'Us Pricing Page</h2>
<p>Current price of coffee beans = <strong>$6.56</strong></p>
<p>Price valid for 15 minutes from Wed Jan 13 11:40:02 2010.</p>
</body></html>

>>>
```

Ln: 14 Col: 4

Wird es ausgeführt, liefert das Programm dieses Ergebnis.

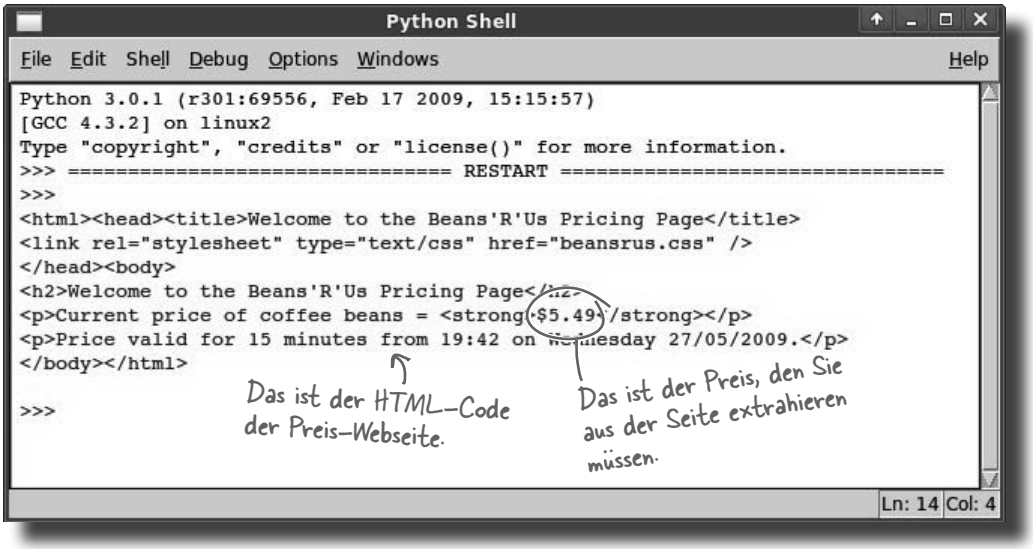
Das sieht wie ein Happen >>rohen<< HTML-Texts aus, dem Format also, das zur Erstellung von Webseiten verwendet wird.

Der Ihnen überlassene Code wendet sich an die Preisseite der Beans'R'Us-Website, um den aktuellen Kaffeebohnenpreis abzufragen. Aber statt einfach nur den Preis auszugeben, gibt er den gesamten HTML-Text für die Webseite selbst aus.

So geht das nicht! Ich will lediglich den aktuellen Kaffeebohnenpreis sehen, nicht auch den ganzen anderen Kram. Glauben Sie, dass Sie mir nur den Preis liefern können?

Der Preis ist in das HTML eingebettet

Schauen Sie sich die Ausgabe des Programms genauer an. Der aktuelle Preis steht genau in der Mitte:

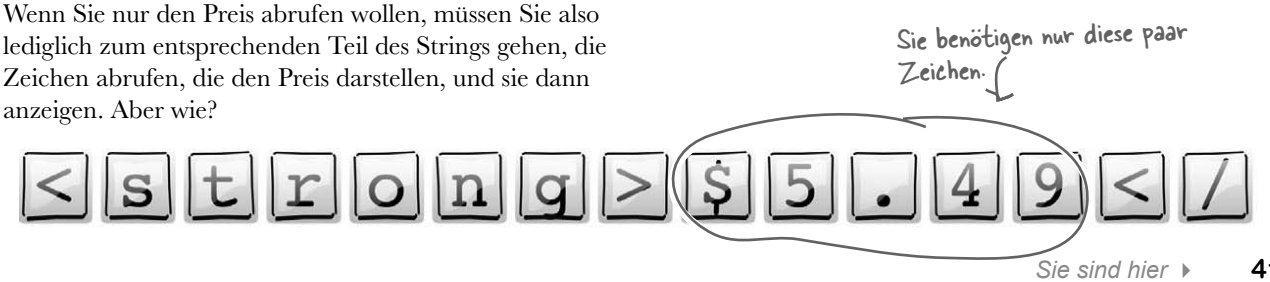


Sie würden die Arbeit des Sternback-Geschäftsführers erheblich erleichtern, wenn Sie für ihn den Bohnenpreis herausziehen und dann allein anzeigen könnten, damit er sich nicht durch das ganze HTML wühlen muss. Aber wie macht man das?

Ein String ist eine Folge von Zeichen

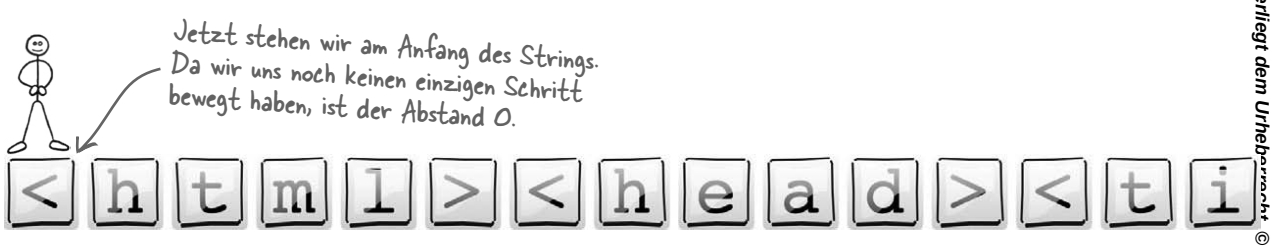
Die Ausgabe des Sternback-Programms ist ein Beispiel für einen **String**. Anders gesagt, es ist eine Zeichenfolge wie diese:

Irgendwo in diesem String steckt der Preis der Kaffeebohnen. Wenn Sie nur den Preis abrufen wollen, müssen Sie also lediglich zum entsprechenden Teil des Strings gehen, die Zeichen abrufen, die den Preis darstellen, und sie dann anzeigen. Aber wie?

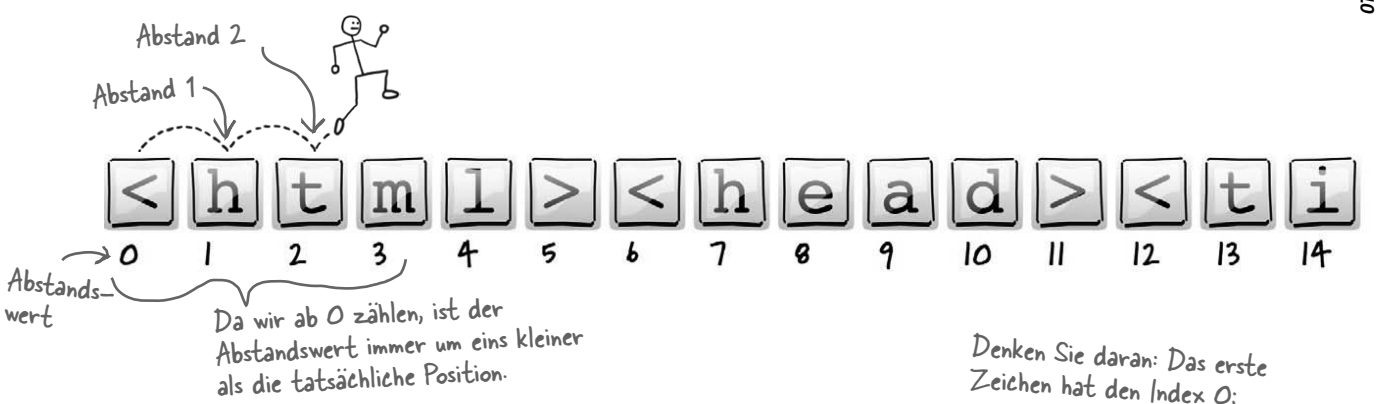


Zeichen in Text finden

Einzelne Zeichen in einem solchen String werden mithilfe von **zwei** Informationen festgehalten: dem **Anfang** des Strings und dem **Abstand** der einzelnen Zeichen zu diesem Anfang. Der Abstand gibt also an, *wie weit* das jeweilige Zeichen vom Anfang des Strings entfernt ist.

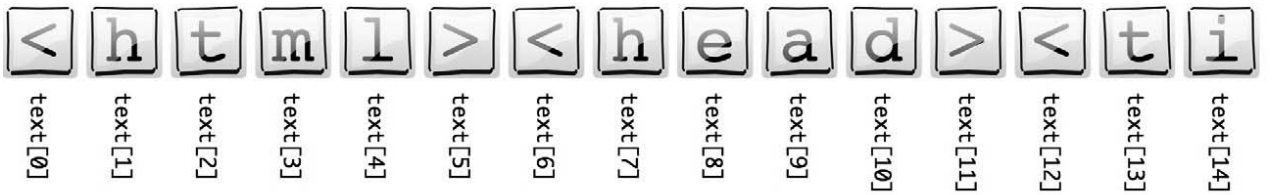


Das **erste** Zeichen eines Strings hat den **Abstand** 0, da es null Zeichen vom Anfang entfernt ist. Das **zweite** Zeichen hat den **Abstand** 1 und so weiter:



Der Abstandswert ist immer 1 kleiner als die Position. Python ermöglicht Ihnen, ein einzelnes Zeichen aus einem String zu lesen, indem Sie nach dem Variablennamen in eckigen Klammern einen Abstandswert angeben. Weil der Abstandswert genutzt wird, um ein Zeichen zu finden, wird er als der **Index** des Zeichens bezeichnet:

Denken Sie daran: Das erste Zeichen hat den Index 0; verweisen wir auf ein einzelnes Zeichen, liegen wir also immer um ein Zeichen daneben.



Aber wie erhält man mehr als ein Zeichen?

Für den Sternback-Auftrag benötigen Sie aber nicht bloß ein einzelnes Zeichen. Sie müssen den gesamten Preis aus einem String mit HTML-Text herausziehen, und dieser Preis besteht aus mehreren Zeichen.

Sie müssen also einen kleineren **Teilstring** aus einem größeren String herausziehen. Ein Teilstring (oder Substring) ist eine Folge von Zeichen, die in einem anderen String enthalten ist. In Python kann man Teilstrings auf ähnliche Weise angeben, wie man einzelne Zeichen aus einem String liest: Anstelle eines einzelnen Indexwerts gibt man in den eckigen Klammern zwei Indexwerte an:

Geben Sie nach dem Variablennamen nur einen Index an, erhalten Sie ein einzelnes Zeichen.

```
s[14]
```

Das liest einen kleineren Teilstring aus dem gesamten String, den >>s<< enthält.

```
s[138:147]
```

Geben Sie zwei Indexwerte an, ziehen Sie eine Gruppe von Zeichen aus dem String, die mit dem ersten Index beginnt und beim zweiten endet (diesen aber nicht einschließt).

Spitzen Sie Ihren Bleistift

Versuchen wir, herauszufinden, was die nachfolgenden Teilstring-Angaben jeweils bedeuten. Stellen Sie sich vor, die Variable `s` sei auf den unten stehenden String gesetzt. Sie sollen ermitteln, was die einzelnen Teilstrings jeweils liefern.

D	e	r	_	W	i	l	d	e	_	i	m	_	W	a	l	d	e	!

`s[4:8]` ……..

`s[10:12]` ……..

`s[13:17]` ……..

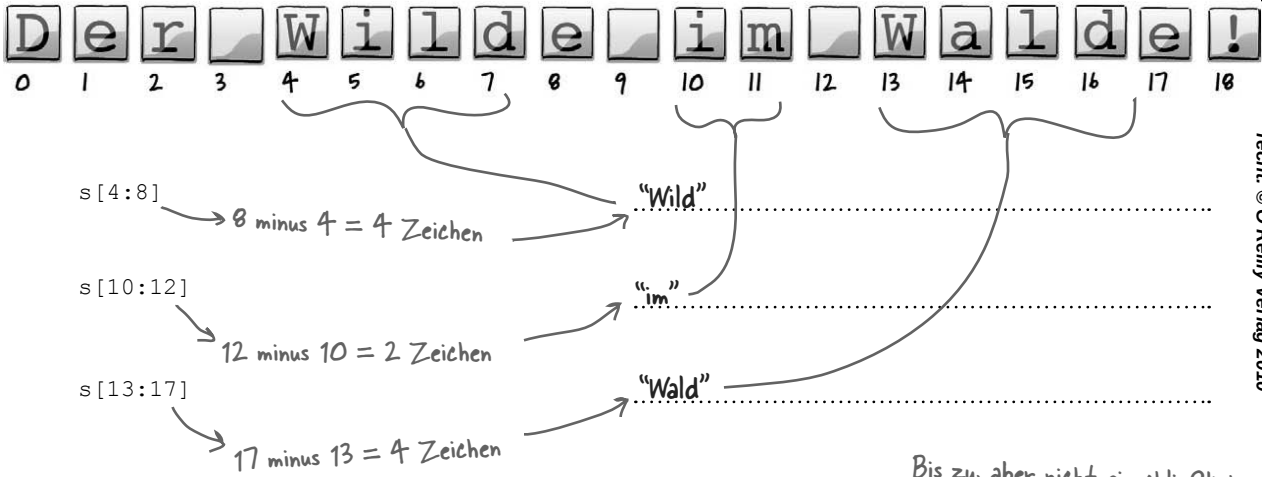
Halten Sie fest, was >>a<< und >>b<< hier repräsentieren.

Formulieren Sie allgemein, was Sie angeben, wenn Sie `s[a:b]` verwenden:

a ist …….. **b** ist …….

Spitzen Sie Ihren Bleistift
Lösung

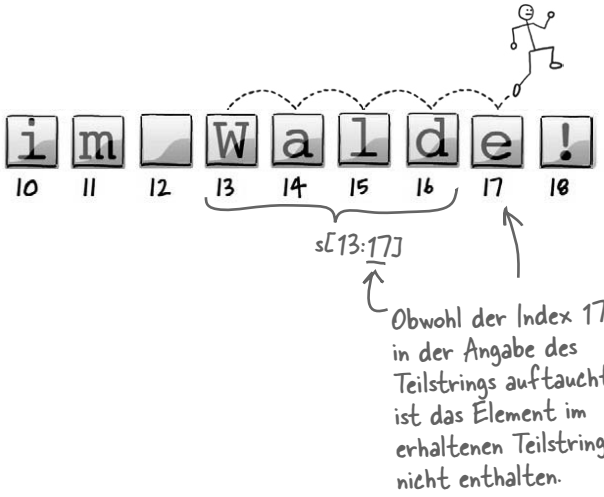
Sie sollten davon ausgehen, dass die Variable `s` auf den unten stehenden String gesetzt ist, und ermitteln, was die Teilstring-Angaben jeweils liefern.



Formulieren Sie allgemein, was Sie angeben, wenn Sie `s[a:b]` verwenden:

Bis zu, aber nicht einschließlich.

a ist der Index des ersten Zeichens . . . . . **b** ist der Index nach dem letzten Zeichen



Der zweite Indexwert gibt den Index des Zeichens nach dem letzten Zeichen des Teilstrings an.

Und das, obwohl der erste Indexwert das Startzeichen des Teilstrings angibt.

Spitzen Sie Ihren Bleistift

Denken Sie daran: Das 235. Zeichen des Strings hat den Index 234.

Sie müssen das Programm also so aktualisieren, dass der Preis herausgezogen wird, der beim 235. Zeichen des Strings beginnt. Der Preis umfasst vier Zeichen. Speichern Sie den Preis-Teilstring in einer Variablen namens `preis`. Notieren Sie unten die neue Version des Programms:

```
import urllib.request

seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices.
html")
text = seite.read().decode("utf8")

print(text)
```

Das ist die aktuelle Version des Codes.

Schreiben Sie hier den neuen Code hin:

Spitzen Sie Ihren Bleistift
Lösung

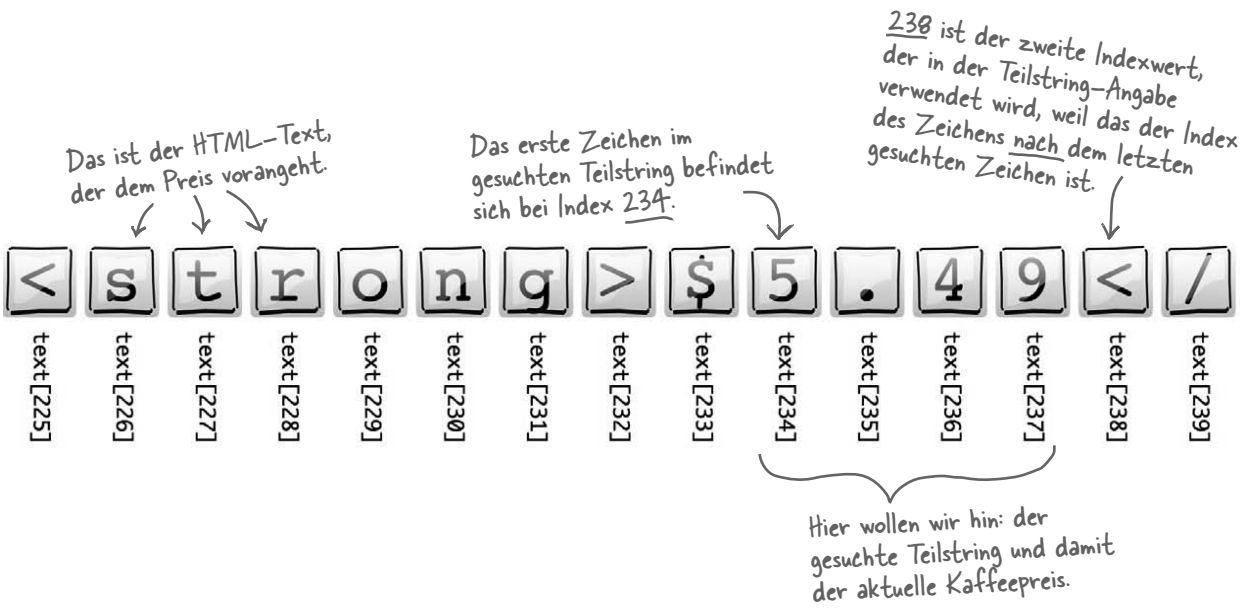
Sie sollten das Programm aktualisieren, um den beim 235. Zeichen beginnenden Teilstring mit dem Preis herauszuziehen. Der Preis umfasst vier Zeichen und soll in einer Variablen namens `preis` gespeichert werden.

```
import urllib.request
seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices.html")
text = seite.read().decode("utf8")
preis = text[234:238]
print(preis)
```

Sie speichern den Teilstring in der Variablen >>preis<<.

Statt >>text<< (die gesamte Webseite) geben Sie jetzt nur >>preis<< (den herausgezogenen Teilstring) aus.

Der Teilstring schließt die Zeichen an den Indizes 234, 235, 236 und 237 ein. Beachten Sie, wie Sie den Teilstring mithilfe der zwei durch ein >>:<< getrennten Indexwerte angegeben haben.



PROBEFAHRT

Geben Sie den Code in IDLE ein, speichern Sie das Programm (wählen Sie einen Namen, der für Sie sinnvoll ist) und führen Sie es aus.

bohnenpreis2.py – /Code/Kapitel2/bohnenpreis2.py

```
import urllib.request

seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices.html")
text = seite.read().decode("utf8")

preis = text[234:238]
print(preis)
```

Python Shell

```
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
5.23
>>>
```

Geschafft: Der aktuelle Kaffeepreis von der Beans'R'Us-Website.

Ln: 7 Col: 4

Das sieht doch schon viel übersichtlicher aus. Statt den gesamten HTML-Text der Webseite auszugeben, haben Sie sie auf den Textausschnitt (Teilstring) beschränkt, den Sie benötigen.

String im Gespräch

Interview der Woche:
Wir fragen String, wie man sich fühlt – als begehrtester Datentyp der Welt.

Von Kopf bis Fuß: Es ist äußerst freundlich von Ihnen, dass Sie sich die Zeit für ein Gespräch mit uns nehmen.

String: Aber nein doch, ganz meinerseits. Nehmen Sie doch Platz, bitte. Ein kleiner Snack gefällig?

Von Kopf bis Fuß: Nein, vielen Dank. Ich frage mich, womit ich beginnen soll? Sie sind ja auf der ganzen Welt für Ihre Leistungen berühmt. Sie, der bereits die Werke von Shakespeare, Goethe und ...

String: Dan Brown.

Von Kopf bis Fuß: ... all die anderen Wunder der Weltliteratur transportiert hat. Und auch alltägliche Dinge wie Namen und Adressen. Verraten Sie uns doch, was Sie so beliebt gemacht hat.

String: Man muss eben Zeichen setzen, besser gesagt, bündeln. Wissen Sie, bevor es mich gab, hielten Computersysteme Text Zeichen für Zeichen nach.

Von Kopf bis Fuß: Das hört ich ziemlich umständlich an.

String: Umständlich? Es war eine Höllenqual.

Von Kopf bis Fuß: Wir verstehen.

String: Vor meiner Zeit glich die Arbeit mit Text einer Radtour ohne Sattel.

Von Kopf bis Fuß: In welcher Weise?

String: Na ja, irgendwann kam man ans Ziel, aber der Weg dahin war eine Tortur.

Von Kopf bis Fuß: Sie machen die Sache also einfacher.

String: Ganz sicher. Man muss nicht mehr Hunderte, Tausende oder noch mehr einzelne Zeichen verwalten, sondern nur noch eine Sache. Mich!

Von Kopf bis Fuß: Das ist natürlich leichter.

String: Ich betrachte mich gern als Agenten für all die Zeichen, mit denen ich arbeite.

Von Kopf bis Fuß: Man arbeitet mit Ihnen und muss sich deswegen nicht mehr mit den einzelnen Zeichen im Speicher herumschlagen.

String: Genau. Ich bin ein Organisator. Ich kümmere mich um die ganzen alltäglichen Zeichengeschäfte. Muss ich kürzer oder länger sein, sorge ich dafür, dass der Platz für die Zeichen verfügbar gemacht wird.

Von Kopf bis Fuß: Können Sie mir etwas über Ihre Teile verraten, diese Teilstrings.

String: Meine Teilstrings, ja. Kleine Späne, die man aus einem Block gehauen hat. Dass ein einfacher Datentyp wie ich so viel Anerkennung erfährt! .

Von Kopf bis Fuß: Ein Taschentuch?

String: Vielen Dank. <Wischt sich die Augen.> Die Jungs stehen mir so nah. Hier, ich habe ein Bild. Sehen Sie die Ähnlichkeit?

Von Kopf bis Fuß: Aber, die sehen ja genau so aus wie Sie ...

String: Vollkommen richtig! Meine Zeichenfolge von 137 bis 149. Genau. Genau wie der Vater. Etwas kürzer. Hat aber noch mehr Haare.

Von Kopf bis Fuß: Teilstrings sind also auch einfach nur Strings.

String: Ja. Strings genau wie ich. Und eines Tages, hoffe ich, werden sie vielleicht ihre eigenen Teilstrings zeugen.

Von Kopf bis Fuß: Trotzdem finden manche Leute das mit Ihren Indizes etwas verwirrend.

String: Was soll man dazu sagen? Ich habe bei null angefangen!

Von Kopf bis Fuß: Wir danken Ihnen, String.

String: War mir ein Vergnügen. Und Sie sind sicher, dass Sie nicht doch etwas essen wollen?

Es gibt keine Dummen Fragen

F: Kann ich jede beliebige Webadresse in diesen Code packen und so die entsprechende Webseite aus dem Internet abrufen?

A: Ja, das geht. Sie können es gern selbst ausprobieren.

F: Brauche ich nicht einen Webbrowser, um Webseiten zu betrachten?

A: Ja, wenn Sie sie in ihrem formatierten Glanz betrachten wollen – mit eingebetteten Bildern, Musik, Videos und Ähnlichem –, ist ein Browser erforderlich. Aber solange Sie nur das HTML brauchen, ist ein Browser überflüssig.

F: Was macht diese `import`-Zeile?

A: Sie gibt dem Programm die Möglichkeit, mit dem Internet zu reden. Der `urllib.request`-Code wird in Python 3 standardmäßig mitgeliefert.

F: Und dieser `urlopen()`-Aufruf öffnet eine Webseite und ruft sie ab?

A: Genau! Die angegebene Webadresse (oder »URL« in korrektem Web-Jargon) wird durch den Aufruf von `urlopen()` abgerufen und zurückgeliefert. Und dass, was sie liefert, wird hier der Variablen `seite` zugewiesen.

F: Und das `urllib.request`-Ding?

A: Das sagt dem Programm nur, dass es die `urlopen()`-Funktion nutzen soll, die standardmäßig mit der Python 3-Technologie zum Lesen von Webseiten ausgeliefert wird. Zu `urllib.request` werden wir später noch etwas mehr sagen. Für den Augenblick sollten Sie sich einfach freuen, dass Sie den Code zum Abrufen von Webseiten nicht selbst schreiben müssen.

F: Ich verstehe, dass der Aufruf von `read()` die Webseite aus der Variablen `seite` liest, aber was macht dieses `decode("utf8")`?

A: Wenn die Webseite aus dem Internet abgerufen wird, befindet sie sich in einem »rohen« Textformat, das für Menschen nicht leicht lesbar ist. Der Aufruf von `decode()` wandelt die Webseite in etwas um, das für das menschliche Auge angenehmer ist.

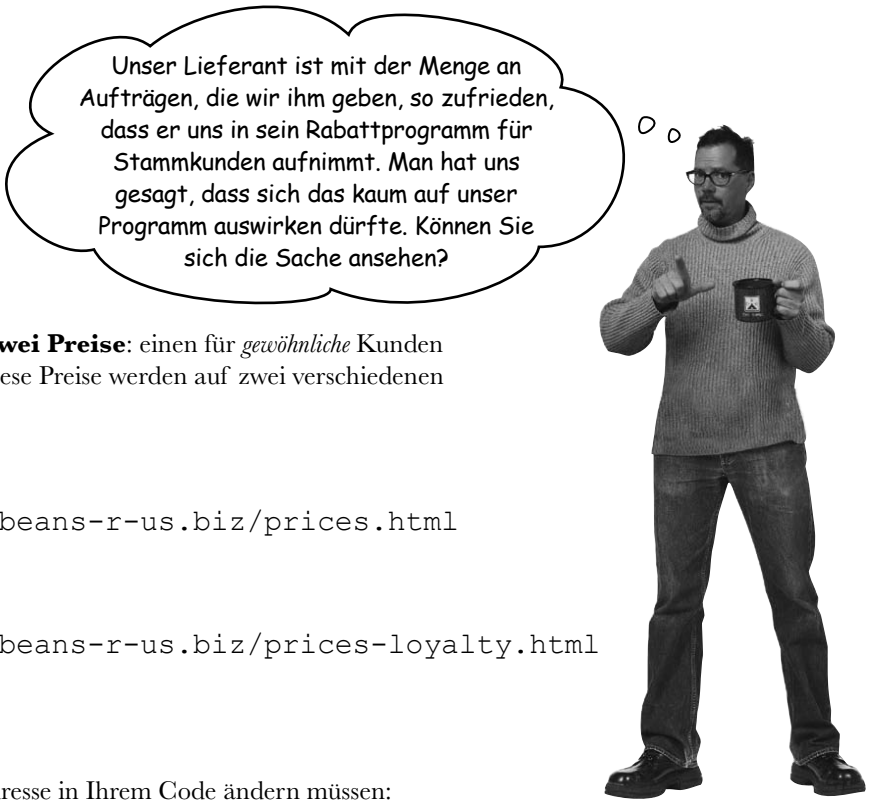
Wenn Sie sich das ansehen wollen, können Sie ja versuchen, den Aufruf von `decode()` aus dem Programm zu entfernen und es dann erneut auszuführen. Sieht etwas seltsam aus, nicht wahr? (Vergessen Sie nicht, den `decode()`-Aufruf wieder einzusetzen, bevor Sie fortfahren.)

Punkt für Punkt

- Sie können das HTML einer Webseite in Textform als **String** herunterladen.
- Eine String ist eine *Folge von Zeichen*.
- Auf einzelne Zeichen in einem String können Sie über einen **Abstand** zugreifen.
- Den Abstand bezeichnet man als **Indexwert** eines Zeichens (oder kurz **Index**).
- Strings in Strings bezeichnet man als **Teilstrings** (oder Substrings).
- Teilstrings werden über zwei Indexwerte angegeben – beispielsweise: `text[10:20]`.
- Der erste Indexwert ist der Ort des ersten Zeichens des Teilstrings.
- Der zweite Indexwert ist der Ort **nach** dem letzten Zeichen des Teilstrings (bis zu, aber nicht einschließlich).
- Ziehen Sie den ersten Index vom zweiten ab, um zu sehen, wie lang der Teilstring sein soll.

Beans'R'Us belohnt Stammkunden

Der Geschäftsführer hat gerade eine erfreuliche Nachricht von seinem Bohnenlieferanten erhalten.



Der Lieferant hat eigentlich **zwei Preise**: einen für *gewöhnliche* Kunden und einen für *Stammkunden*. Diese Preise werden auf zwei verschiedenen Webseiten veröffentlicht:

Normale Kunden erhalten den Preis hier.

`http://www.beans-r-us.biz/prices.html`

Stammkunden erfahren Ihren Preis hier.

`http://www.beans-r-us.biz/prices-loyalty.html`

Das heißt, dass Sie die Webadresse in Ihrem Code ändern müssen:

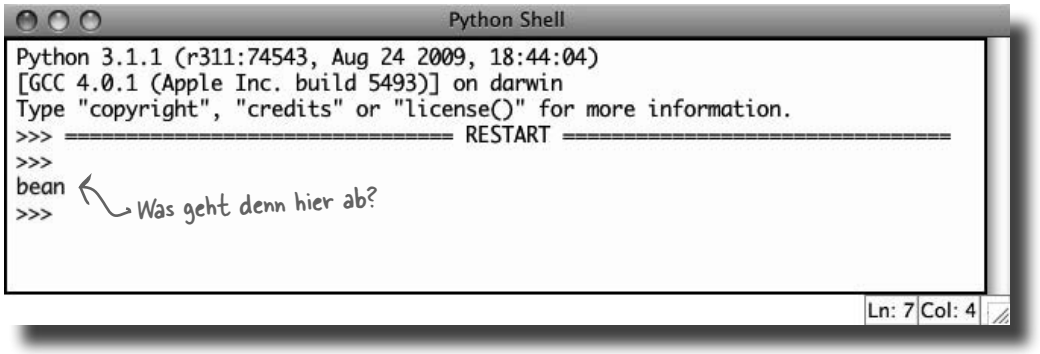
```
import urllib.request

seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices-loyalty.html")
text = seite.read().decode("utf8")

preis = text[234:238]
print(preis)
```

Das ist die neue Adresse.

Schauen wir, ob alles korrekt läuft.

PROBEFAHRT

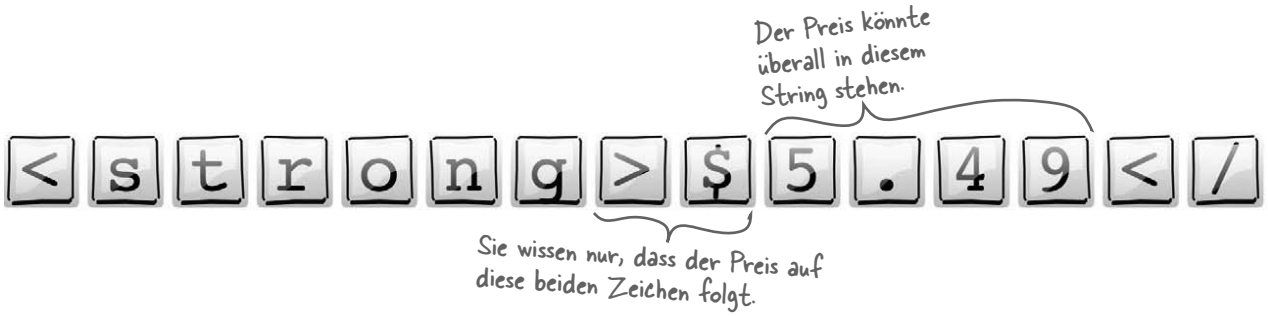
Als Sie das Programm ausführten, mussten Sie sich enttäuscht **dieses** Ergebnis ansehen ...



Es wird kein **Preis mehr angezeigt**. Was ist da passiert?

Der Preis steht an einer anderen Stelle

Die Webseite für Stammkunden ist erheblich **dynamischer** als die alte Webseite. Die Webseite für gewöhnliche Kunden zeigt den Preis immer in dem Teilstring an, der bei Index 234 beginnt. Auf der Webseite für Stammkunden ist das nicht der Fall. Der Preis kann fast überall stehen. Sicher wissen Sie nur, dass der Preis auf den Teilstring **>$** folgt:



Sie müssen nach dem String mit dem Preis suchen.

Suchen ist nicht leicht

Sie wissen bereits, wie man einen Teilstring findet, könnten also die gesamte Webseite durchlaufen und jeweils zwei Zeichen darauf prüfen, ob sie gleich >$ sind:

Man *könnte* es auf diese Weise tun ... aber *sollte* man das auch?

Man muss sich bei dieser Operation um eine Menge Dinge kümmern. Welche Zeichen man aktuell vergleicht. An welcher Position des Strings man sich gerade befindet. Und was passiert, wenn »>$« nicht gefunden wird. In einem String nach Teilstrings zu suchen, ist offensichtlich doch um einiges komplizierter, als es zunächst den Anschein hatte ...

Aber was können Sie sonst noch tun, wenn Sie Code schreiben müssen, der einen String in einem String sucht?

Python-Daten sind schlau

Je mehr Code Sie schreiben, umso häufiger werden Sie feststellen, dass Sie immer wieder die gleichen Dinge mit den Daten in Ihren Variablen machen. Damit Sie nicht ständig den gleichen Code schreiben müssen, bieten Programmiersprachen **eingebaute Funktionalitäten**. Python-Daten sind *schlau*: Sie können **Dinge tun**.

Schauen wir uns ein Beispiel an.

Stellen Sie sich vor, Sie hätten eine Variable mit Textdaten, die Sie in Großbuchstaben anzeigen wollen:

```
text = "Schneckenrennen, Montag 4 Uhr."
```

Sie *könnten* Code schreiben, der die Zeichen des Strings einzeln durchläuft und jeweils den passenden Großbuchstaben ausgibt. Aber da Sie mit einer Programmiersprache wie Python arbeiten, können Sie stattdessen einfach Folgendes machen:

Der Punkt heißt: »Ruf die Methode auf«, und zwar der angegebenen Variablen.

»upper()« ist eine String-Methode.

```
print(text.upper())

SCHNECKENRENNEN, MONTAG 4 UHR.
```

Das wird ausgegeben, wenn der Wert der Variablen »text« in Großbuchstaben angefordert wird.

Aber was heißt `text.upper()`?

Also, `text` ist der String, der unseren Text enthält. Das nachfolgende `.upper()` bezeichnet man als eine String-**Methode**. Eine Methode ist einfach eine Anweisung für einen String. Rufen Sie `text.upper()` auf, sagen Sie dem String, dass er eine Version seiner Daten liefern soll, in der alle Zeichen in Großbuchstaben stehen.

Aber gibt es auch eine String-Methode, die uns beim Suchen von Strings in Strings helfen kann?

Wer macht was?

Hier sehen Sie einige der eingebauten String-Methoden, die Python mitbringt. Ordnen Sie die Methoden den passenden Beschreibungen zu. Eine haben wir bereits für Sie erledigt.

Methode	Was die Methode macht
text.endswith(".jpg")	Liefert eine Kopie des Strings, in dem alle Vorkommen eines Teilstrings durch einen anderen ersetzt werden.
text.upper():	Liefert eine Kopie des Strings, die alle Zeichen als Kleinbuchstaben enthält.
text.lower():	Liefert den Wert **True**, wenn der String-Anfang dem angegebenen Teilstring entspricht.
text.replace("morgen", "Dienstag"):	Liefert den Wert **True**, wenn das String-Ende dem angegebenen Teilstring entspricht.
text.strip():	Liefert den ersten Indexwert, wenn der angegebene Teilstring gefunden wird.
text.find("python"):	Liefert eine Kopie des Strings, aus dem Leerraumzeichen am Anfang und am Ende entfernt wurden.
text.startswith("<HTML>")	Liefert eine Kopie des Strings, die alle Zeichen des Strings in Großbuchstaben enthält.

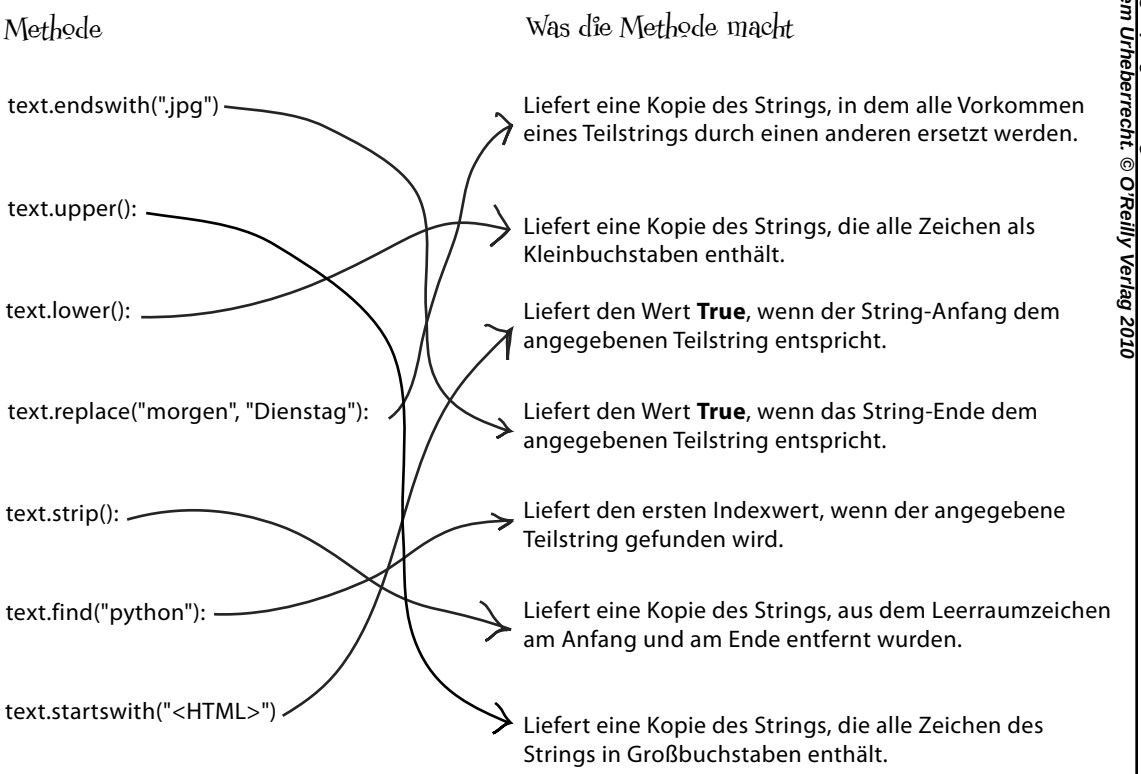
Spitzen Sie Ihren Bleistift

Welche der oben aufgeführten Methoden würden Sie einsetzen, um den Preis im HTML-Text für die Beans'R'Us-Webseite zu finden?

.....

WER MACHT WAS? LÖSUNG

Hier sehen Sie einige der eingebauten String-Methoden, die Python mitbringt. Sie sollten die Methoden den passenden Beschreibungen zuordnen.



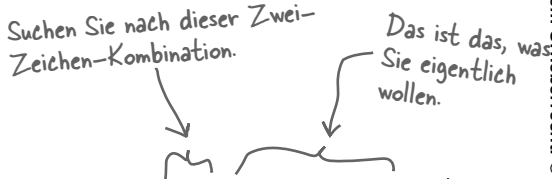
Spitzen Sie Ihren Bleistift Lösung

Welche der oben aufgeführten Methoden würden Sie einsetzen, um den Preis im HTML-Text für die Beans'R'Us-Webseite zu finden?

Die Methode >>find()<<.

Sie müssen das Programm zur Preisermittlung aktualisieren, damit es den aus vier Zeichen bestehenden Teilstring herauszieht, der auf das Vorkommen der Zeichen »>$« folgt. Schreiben Sie die neue Version des Codes in die leeren Zeilen unten.

Tipp: Vergessen Sie nicht, dass `find()` die *Anfangsposition* eines Substrings liefert. Haben Sie »>$« gefunden, können Sie Pythons **Additionsoperator** nutzen, um die Stelle zu berechnen, an der Sie den Teilstring herausziehen müssen. Der Additionsoperator ist das Pluszeichen (+).




```
coffee beans = <strong>$5.49</
strong></p><p>Price valid for
```

LÖSUNG ZUR ÜBUNG

Sie sollten das Programm zur Preisermittlung aktualisieren, damit es den aus vier Zeichen bestehenden Teilstring herauszieht, der auf das Vorkommen der Zeichen »>$« folgt.

Tipp: Vergessen Sie nicht, dass `find()` die *Anfangsposition* eines Substrings liefert. Haben Sie »>$« gefunden, können Sie Pythons **Additionsoperator** nutzen, um die Stelle zu berechnen, an der Sie den Teilstrings herausziehen müssen. Der Additionsoperator ist das Pluszeichen.

Dieser Code bleibt unverändert.

```
import urllib.request

seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices-loyalty.html")
text = seite.read().decode("utf8")

wo = text.find('>$')

preisanfang = wo + 2
preisende = preisanfang + 4

preis = text[preisanfang:preisende]

print(preis)
```

Nach der Anfangsposition der >$-Zeichenfolge suchen.

Das ist der Additionsoperator.

Der Preis beginnt zwei Positionen weiter hinten im String, sein Ende noch folgt einmal 4 Positionen nach dem Anfang.

Haben Sie daran gedacht, den Preis auszugeben, nachdem Sie ihn gefunden haben?

Kennen wir die Indexwerte für Anfang und Ende, ist es kein Problem mehr, den gesuchten Teilstring anzugeben.

PROBEFAHRT

Jetzt müsste Ihr Programm den Preis finden können, egal wo er auf der Seite erscheint.

bohnenpreis3.py – /Code/Kapitel2/bohnenpreis3.py

```
import urllib.request

seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices-loyalty.html")
text = seite.read().decode("utf8")

wo = text.find(">$")

preisanfang = wo + 2
preisende = preisanfang + 4

preis = text[preisanfang:preisende]

print(preis)
```

Version 3 des Programms.

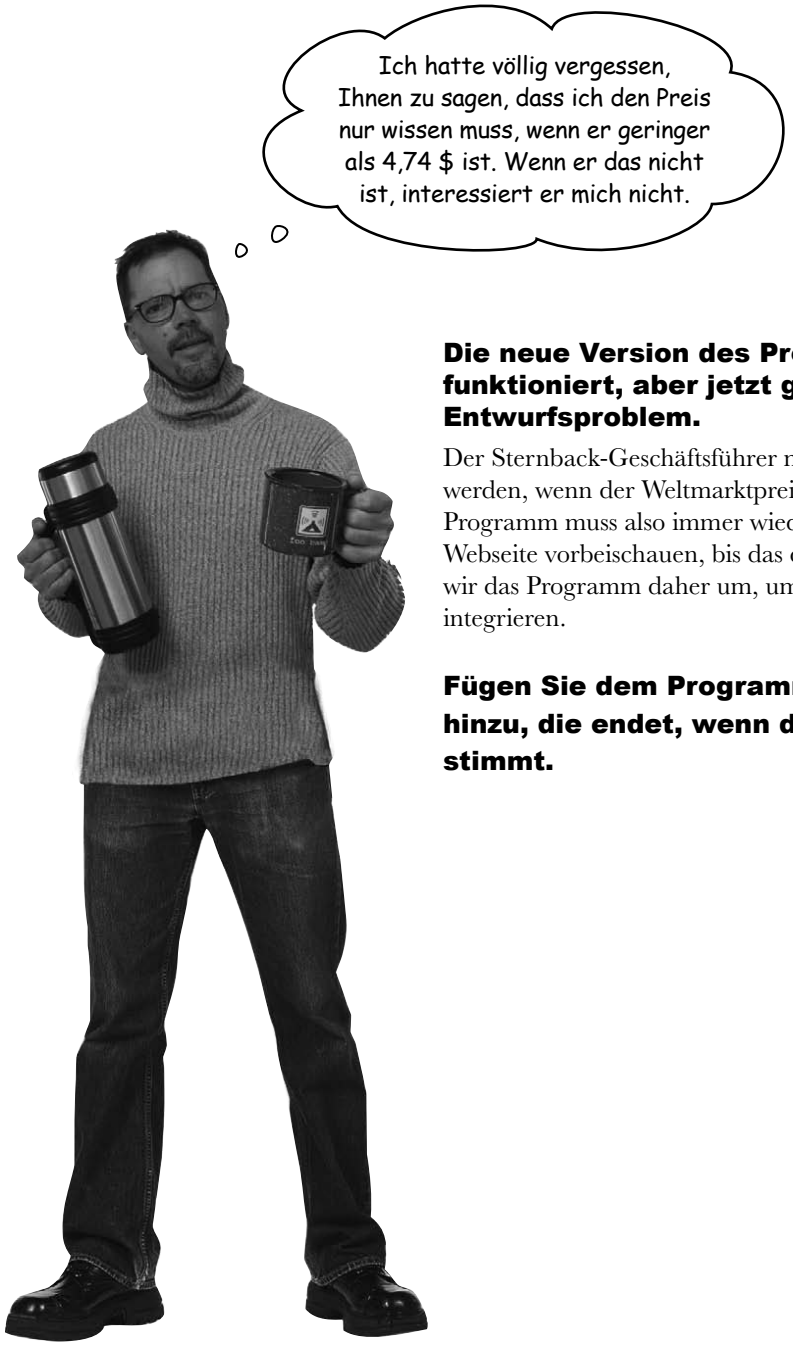
Python Shell

```
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
4.69
>>>
```

Ln: 7 Col: 4

Der aus dem umfangreicheren HTML-String herausgezogene Preis.

Es funktioniert! Durch die Ergänzung von ein klein wenig zusätzlichem Code haben wir das Programm erheblich intelligenter und gleichzeitig auch nützlicher gemacht.



Die neue Version des Programms funktioniert, aber jetzt gibt es ein Entwurfsproblem.

Der Sternback-Geschäftsführer möchte informiert werden, wenn der Weltmarktpreis unter 4,74 $ fällt. Das Programm muss also immer wieder bei der Beans'R'Us-Webseite vorbeischauen, bis das der Fall ist. Strukturieren wir das Programm daher um, um diese Funktionalität zu integrieren.

Fügen Sie dem Programm eine Schleife hinzu, die endet, wenn der Kaffeepreis stimmt.

Code-Magneten

Der Programmcode für die neue Funktionalität ist vollkommen durcheinandergeraten. Ordnen Sie die Magneten so, dass das Programm eine Schleife ausführt, bis der Preis kleiner oder gleich 4,74 $ ist.

```
text = seite.read().decode("utf8")
```

```
preis = 99.99
```

```
preis = text[preisanfang:preisende]
```

```
import urllib.request
```

```
preisanfang = wo + 2
```

```
wo = text.find('>$')
```

```
print ("Kaufen!")
```

```
preisende = preisanfang + 4
```

```
while preis > 4.74:
```

```
seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices-loyalty.html")
```

Code-Magneten, Lösung

Der Programmcode für die neue Funktionalität ist vollkommen durcheinandergeraten. Ordnen Sie die Magneten so, dass das Programm eine Schleife ausführt, bis der Preis kleiner oder gleich 4,74 $ ist.

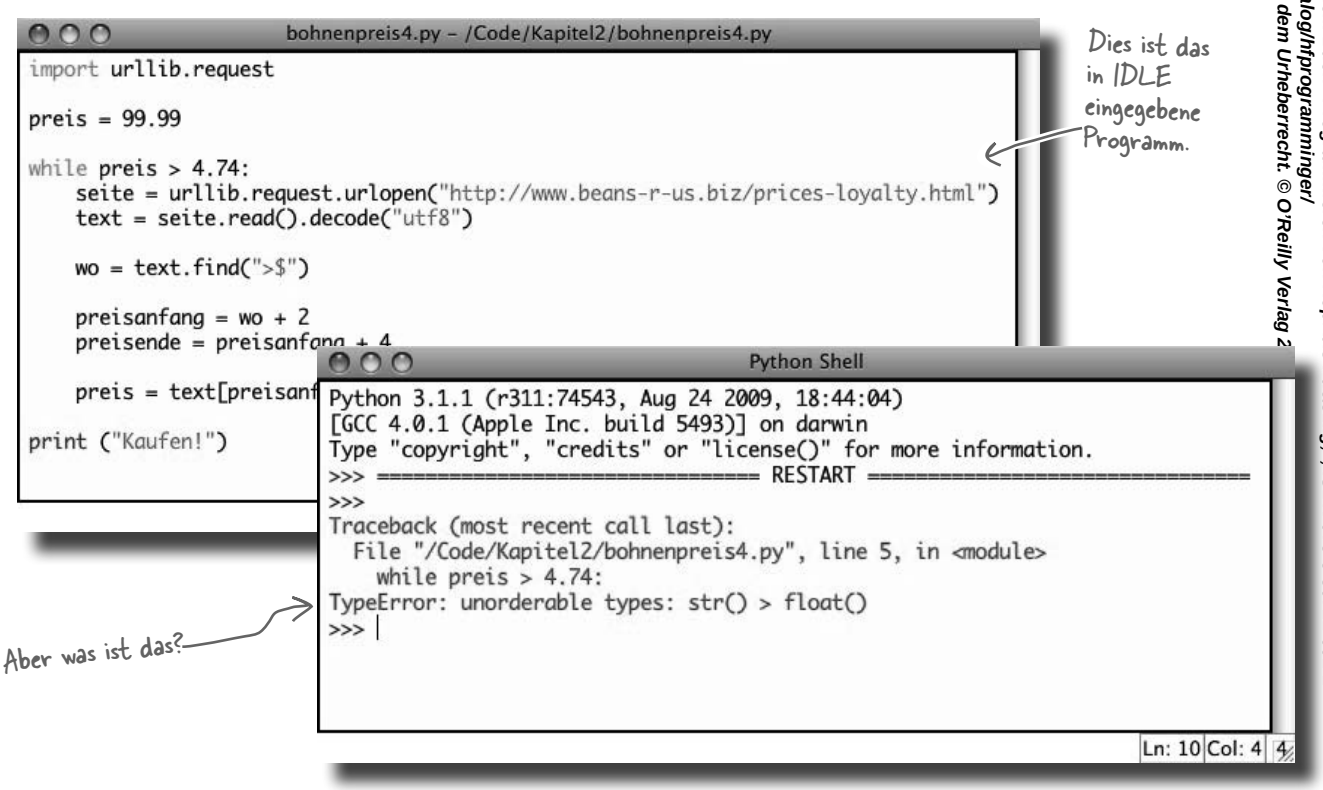
```
import urllib.request

preis = 99.99

while preis > 4.74:

    seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices-loyalty.html")

    text = seite.read().decode("utf8")

    wo = text.find('>$')

    preisanfang = wo + 2

    preisende = preisanfang + 4

    preis = text[preisanfang:preisende]

print ("Kaufen!")
```

Haben Sie daran gedacht, diese Zeilen einzurücken? Sie stehen innerhalb der Schleife.

Diese Zeile darf nicht eingerückt sein, da sie außerhalb der Schleife steht.

PROBEFAHRT

Geben Sie die neue Version des Programmcodes in ein IDLE-Bearbeitungsfenster ein und führen Sie sie aus.



Scheint, als wäre mit unserem Programm etwas schiefgelaufen. Was heißt `TypeError`? Und was ist passiert?

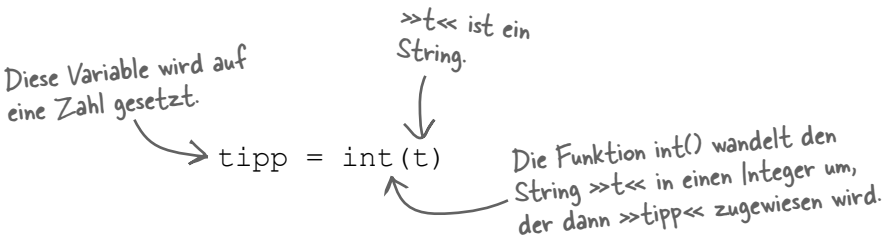
KOPF-NUSS

Schauen Sie sich die Fehlermeldung genau an. Versuchen Sie, die Codezeile zu identifizieren, die den Absturz verursacht hat, und überlegen Sie, was ein `TypeError` sein könnte. Warum ist der Code wohl abgestürzt?

Strings und Zahlen sind unterschiedlich

Das Programm ist abgestürzt, weil wir versuchten, einen **String** mit einer **Zahl** zu vergleichen. Das ist etwas, was für viele Programmiersprachen eine ausgesprochen unverständliche Operation ist. Wenn wir bestimmte Daten als Strings oder Zahlen bezeichnen, geht es uns nicht nur um den *Inhalt* der Variablen. Wir verweisen damit auch auf ihren **Datentyp**. Wenn zwei Datenhappen *unterschiedliche Typen* haben, können wir sie nicht vergleichen.

Denken Sie an das letzte Kapitel zurück. In unserem Ratespiel mussten wir uns mit diesem Problem bereits auseinandersetzen:



Im Ratespiel-Programm mussten Sie den Tipp des Benutzers mithilfe der Funktion `int()` in einen **Integer** (eine ganze Zahl) *umwandeln*. Aber der Preis für die Kaffeebohnen ist *keine* ganze Zahl, da er Nachkommastellen enthält. Er ist eine **Fließkommazahl** oder kurz **Float**. Wenn Sie einen String in einen Float umwandeln müssen, brauchen Sie eine andere Funktion als `int()`, und das ist die Funktion `float()`:

Präziser wäre >>Fließpunktzahl<<, da Python (wie alle anderen Programmiersprachen auch) einen Punkt als Dezimaltrenner nutzt.

```
float("4.99")
```

Wie int(), arbeitet aber mit Zahlen, die Nachkommastellen enthalten.

PROBEFAHRT

Das müsste sich doch sehr leicht reparieren lassen. Wenn Sie die Funktion `float()` einsetzen, um den Teilstring `preis` umzuwandeln, sollten Sie den Preiswert mit 4.74 vergleichen können:

bohnenpreis4.py – /Code/Kapitel2/bohnenpreis4.py

```
import urllib.request

preis = 99.99

while preis > 4.74:
    seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices-loyalty.html")
    text = seite.read().decode("utf8")

    wo = text.find(">$")

    preisanfang = wo + 2
    preisende = preisanfang + 4

    preis = float(text[preisanfang:preisende])

print ("Kaufen!")
```

Diesmal läuft das Programm ohne Probleme.

Der aktualisierte Code.

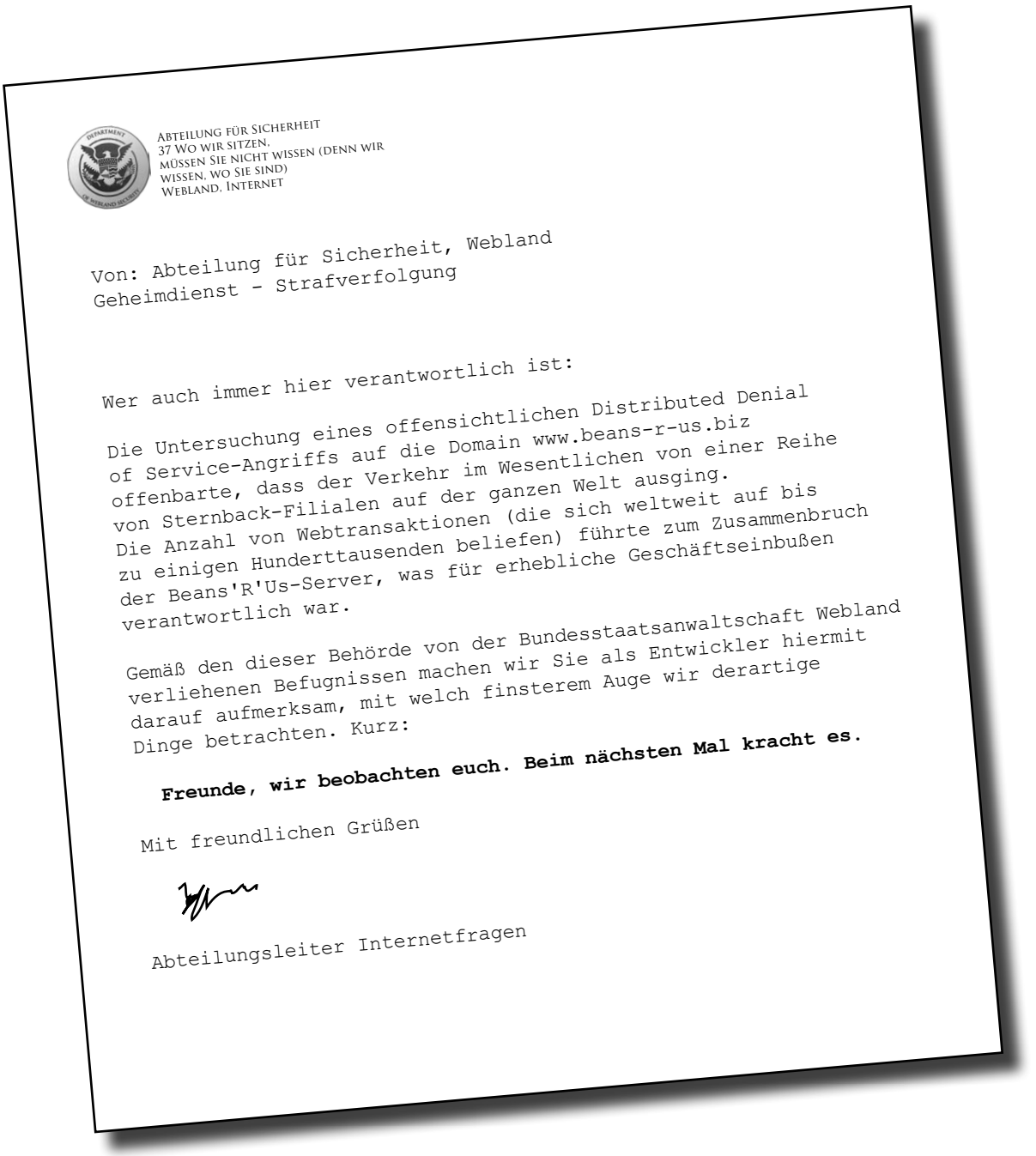
Python Shell

```
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
Kaufen!
>>>
```

Ln: 7 Col: 4

Das ist viel besser. Jetzt wartet Ihr Programm geduldig ab, bis der Preis weit genug fällt, und sagt dem Sternback-Geschäftsführer dann, dass er jetzt beim Kaffeebohnenkauf ein Schnäppchen machen kann.

Klasse! Damit kann ich mich um meine restliche Arbeit kümmern und erfahre nur, wenn der Preis für Kaffeebohnen günstig ist. Das spart uns Millionen! Ich werde alle Filialen **weltweit** anweisen, Ihr Programm einzusetzen.



Abteilung für Sicherheit
37 Wo wir sitzen,
müssen Sie nicht wissen (denn wir
wissen, wo Sie sind)
Webland, Internet

Von: Abteilung für Sicherheit, Webland
Geheimdienst - Strafverfolgung

Wer auch immer hier verantwortlich ist:

Die Untersuchung eines offensichtlichen Distributed Denial of Service-Angriffs auf die Domain www.beans-r-us.biz offenbarte, dass der Verkehr im Wesentlichen von einer Reihe von Sternback-Filialen auf der ganzen Welt ausging. Die Anzahl von Webtransaktionen (die sich weltweit auf bis zu einigen Hunderttausenden beliefen) führte zum Zusammenbruch der Beans'R'Us-Server, was für erhebliche Geschäftseinbußen verantwortlich war.

Gemäß den dieser Behörde von der Bundesstaatsanwaltschaft Webland verliehenen Befugnissen machen wir Sie als Entwickler hiermit darauf aufmerksam, mit welch finsterem Auge wir derartige Dinge betrachten. Kurz:

Freunde, wir beobachten euch. Beim nächsten Mal kracht es.

Mit freundlichen Grüßen

Abteilungsleiter Internetfragen

Das klingt komisch. Was ist passiert?

Das Programm hat den Beans'R'Us-Server überfordert

Sieht so aus, als gäbe es ein Problem mit unserem Programm. Es sendet so viele Anfragen, dass die Beans'R'Us-Website in die Knie geht. Aber warum ist das passiert? Nehmen wir uns unseren Code noch einmal vor:

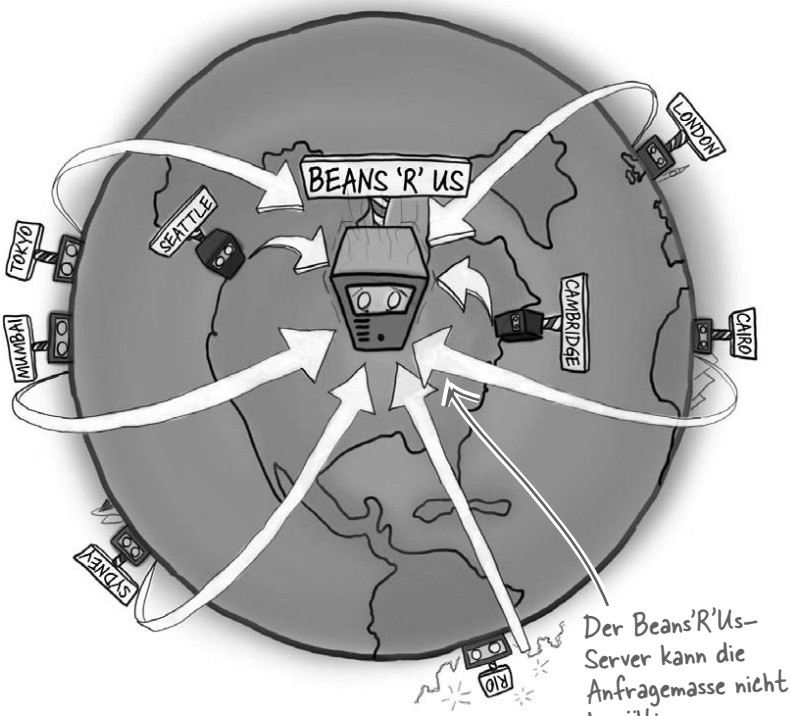
Hier ist der Code in seiner aktuellen Form.

```
import urllib.request
preis = 99.99
while preis > 4.74:
    seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices-loyalty.html")
    text = seite.read().decode("utf8")
    wo = text.find('>$')
    preisanfang = wo + 2
    preisende = preisanfang + 4
    preis = float(text[preisanfang:preisende])
print ("Kaufen!")
```

Ist der Wert von `preis` nicht klein genug (größer als 4.74), springt das Programm *unmittelbar* an den Anfang der Schleife zurück und sendet eine neue Anfrage.

Das heißt, dass das Programm in dieser Form **Tausende** von Anfragen pro Stunde generiert. Multiplizieren Sie das mit der Anzahl von Sternback-Filialen auf der ganzen Welt, beginnen Sie vielleicht, die Größenordnung des Problems zu verstehen:

Sie müssen die Preisanfrage verzögern. Aber wie?



Zeit ... hätten wir nur mehr davon!

Und als Sie sich so langsam völlig verloren fühlten, erhielten Sie einen Telefonanruf von dem Kerl, der für die ursprüngliche Version des Sternback-Programms verantwortlich war:

Offenbar konnte er sich aufgrund eines Sturms in den Bergen nicht mehr melden. Aber jetzt hat er einen Vorschlag. Sie müssen steuern, wie oft eine Anfrage an den Beans'R'Us-Webserver gesendet wird. Eine Möglichkeit dafür bietet eine **Zeit-Bibliothek**. Anscheinend können wir mit ihrer Hilfe nur beispielsweise alle *15 Minuten* eine Anfrage senden. Und das müsste die Serverbelastung mindern.

Bleibt nur eine Frage zu klären: **Was bitte ist eine Bibliothek?**

Sie nutzen bereits eine Bibliothek

Schauen Sie sich die erste Zeile des ursprünglichen Codes an:

```
import urllib.request
```

Das sagt, dass wir Code nutzen werden, der in der Bibliothek »urllib.request« gespeichert ist.

Der urllib.request-Code

Diese `import`-Zeile sagt Python, dass Sie beabsichtigen, Code aus einer Bibliothek zu nutzen, die den Namen `urllib.request` hat. Eine Bibliothek ist vorgefertigter Code, den Sie in Ihren Programmen nutzen können. Der hier genutzte `urllib.request`-Code greift auf Daten im Internet zu. Das ist eine Bibliothek, die *standardmäßig* mit Python ausgeliefert wird.

Wenn Sie sehen wollen, wie der Code verwendet wird, müssen Sie sich diese Zeile ansehen:

```
seite = urllib.request.urlopen("http://...")
```

Bibliotheksname.

Alle Bibliotheken enthalten Funktionen, die Sie in Ihren Programmen nutzen können.

Der auf das =-Zeichen folgende Code ruft eine Funktion in `urllib.request` auf, die `urlopen()` heißt. Beachten Sie, wie wir den Code, den wir nutzen wollen, angegeben haben: `urllib.request` plus ».« und dann den Namen der Funktion.

```
<Bibliotheksname> . <Funktionsname>
```

Aber wie hilft uns diese time- oder Zeit-Bibliothek? Schauen wir mal ...

Lange Übung

Unten sehen Sie einige Funktionen, die von Pythons eingebauter Bibliothek time zur Verfügung gestellt werden:

Python-Bibliotheksdokumentation: time

time.clock()
Die aktuelle Zeit in Sekunden als Fließkommazahl.

time.daylight()
Liefert 0, wenn aktuell nicht Sommerzeit ist.

time.gmtime()
Liefert den aktuellen UTC-Datums- und Zeitwert (ohne Einbeziehung der Zeitzone).

time.localtime()
Liefert die aktuelle lokale Zeit (unter Einbeziehung Ihrer Zeitzone).

time.sleep(secs)
Für eine bestimmte Anzahl an Sekunden nichts tun.

time.time()
Liefert die Anzahl an Sekunden seit dem 1. Januar 1970.

time.timezone()
Liefert die Zeitverschiebung zwischen Ihrer Zeitzone und der UTC-Zeitzone (London).

Eine dieser Funktionen benötigen Sie, um Ihren Code zu reparieren.

Welche ist das? Kreisen Sie die Funktion ein, die Sie wohl benötigen.

Verbessern Sie, nachdem Sie die geeignete Funktion ermittelt haben, den Code, um zu steuern, wie oft eine Anfrage nach der Webseite an den Server gesendet wird. Der Beans'R'Us-Webmaster hat sich mit Ihnen in Verbindung gesetzt und Ihnen mitgeteilt, dass die Preisinformationen auf der Webseite alle 15 Minuten aktualisiert werden. Setzen Sie den erforderlichen Code in die Lücken ein.

Tipp: 15 Minuten entspricht 15 mal 60 Sekunden, d.h. 900 Sekunden. Und: Wenn Sie Funktionalitäten nutzen wollen, die von einer Bibliothek kommen, müssen Sie diese zuvor importieren.

```
import urllib.request

……….

preis = 99.99
while preis > 4.74:
    ……….
    seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices.html")
    text = seite.read().decode("utf8")
    wo = text.find('>$')
    preisanfang = wo + 2
    preisende = preisanfang + 4
    preis = float(text[preisanfang:preisende])
print ("Kaufen!")
```

Lange Übung Lösung

Unten sehen Sie einige Funktionen, die von Pythons eingebauter Bibliothek time zur Verfügung gestellt werden:

Python-Bibliotheksdokumentation: time

time.clock()
Die aktuelle Zeit in Sekunden als Fließkommazahl.

time.daylight()
Liefert 0, wenn aktuell nicht Sommerzeit ist.

time.gmtime()
Liefert den aktuellen UTC-Datums- und Zeitwert (ohne Einbeziehung der Zeitzone).

time.localtime()
Liefert die aktuelle lokale Zeit (unter Einbeziehung Ihrer Zeitzone).

time.sleep(secs)
Für eine bestimmte Anzahl an Sekunden nichts tun.

time.time()
Liefert die Anzahl von Sekunden seit dem 1. Januar 1970.

time.timezone()
Liefert die Zeitverschiebung zwischen Ihrer Zeitzone und der UTC-Zeitzone (London).

Das scheint die Funktion zu sein, die wir brauchen.

Eine dieser Funktionen benötigen Sie, um Ihren Code zu reparieren.

Welche ist das? Sie sollten die Funktion einkreisen, die Sie benötigen.

Sie sollten, nachdem Sie die geeignete Funktion ermittelt hatten, den Code verbessern, um zu steuern, wie oft eine Anfrage nach der Webseite an den Server gesendet wird. Der Beans'R'Us-Webmaster hat sich mit Ihnen in Verbindung gesetzt und Ihnen mitgeteilt, dass die Preisinformationen auf der Webseite alle 15 Minuten aktualisiert werden.

Tipp: 15 Minuten entspricht 15 mal 60 Sekunden, d.h. 900 Sekunden. Und: Wenn Sie Funktionalitäten nutzen wollen, die von einer Bibliothek kommen, müssen Sie diese zuvor importieren.

Importieren Sie zu Anfang des Programms die Bibliothek. Das verschafft dem Programm Zugriff auf alle eingebauten Funktionalitäten, die diese Bibliothek stellt.

```
import urllib.request
import time

preis = 99.99
while preis > 4.74:
    time.sleep(900)
    seite = urllib.request.urlopen("http://www.beans-r-us.biz/prices.html")
    text = seite.read().decode("utf8")
    wo = text.find('>$')
    preisanfang = wo + 2
    preisende = preisanfang + 4
    preis = float(text[preisanfang:preisende])
print ("Kaufen!")
```

Nutzen Sie diese Funktion der Bibliothek, um das Programm zwischen den Anfragen für 15 Minuten schlafen zu legen.

Die Ordnung ist wiederhergestellt

Sternback-Kaffee steht nicht mehr auf der schwarzen Liste, weil das Preisprüfungsprogramm den Beans'R'Us-Webserver nicht mehr zusammenbrechen lässt. Die freundlichen Menschen der Webland-Sicherheit haben sich heimlich und still zurückgezogen.

Kaffeebohnen werden bestellt, wenn der Preis stimmt!

Ihr Programmierwerkzeugkasten

Sie haben Kapitel 2 bewältigt. Schauen wir uns an, was Sie in diesem Kapitel gelernt haben:

Programmierwerkzeuge

* Strings sind Folgen einzelner Zeichen.
* Einzelne String-Zeichen werden über einen Index angesprochen.
* Indizes sind Abstandswerte, die bei null beginnen.
* Methoden sind eingebaute Funktionalitäten von Datentypen.
* Programmierbibliotheken bieten Sammlungen von vorgefertigtem Code.
* Daten in Variablen haben nicht nur einen Wert, sondern auch einen »Datentyp«.
* Number ist ein Datentyp.
* String ist ein Datentyp.

Python-Werkzeuge

* s[4] – greift auf das 5. Zeichen der Variablen »s« (einen String) zu.
* s[6:12] – greift auf einen Teilstring von »s« zu (bis zum zweiten Index, aber nicht einschließlich).
* s.find() ist eine Methode zum Suchen in Strings.
* s.upper() ist eine Methode, die Strings in GROSSBUCHSTABEN umwandelt.
* float() wandelt Strings in Fließkommazahlen um, die als »Floats« bezeichnet werden.
* + ist der Additionsoperator.
* > ist der Größer-als-Operator.
* urllib.request ist eine Bibliothek zur Kommunikation mit dem Internet.
* time ist eine Bibliothek zur Arbeit mit Datum/Zeit.