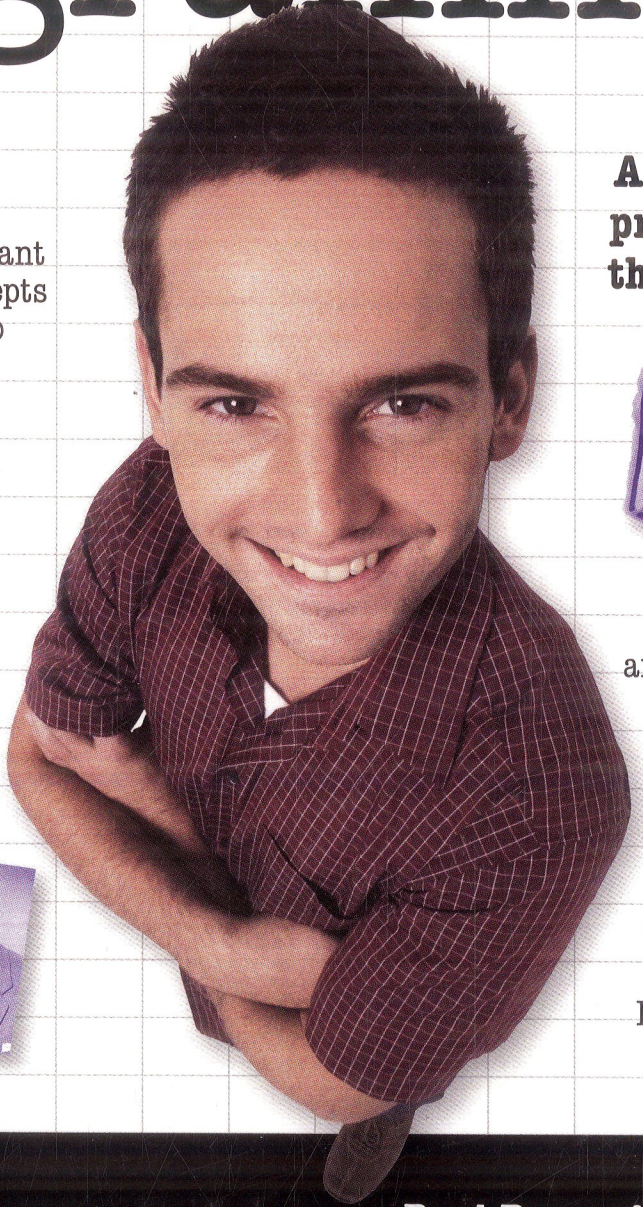
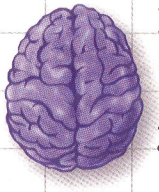
A Brain-Friendly Guide

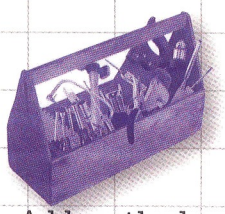
Head First Programming



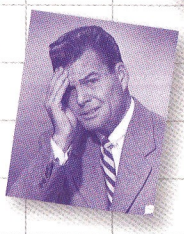
A learner's guide to programming, using the Python language



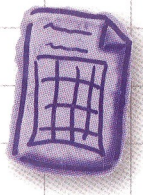
Load important coding concepts straight into your brain



Add methods, functions, and objects to your toolbox



Avoid embarrassing mishaps with input/output



Process your data like a pro

Build a functional and attractive graphical application

Learn how to automate repetitive tasks

O'REILLY®

Paul Barry & David Griffiths

Table of Contents (Summary)

Table of Contents (the real thing)

Intro

Your brain on Programming. Here *you* are trying to *learn* something, while here your *brain* is doing you a favor by making sure the learning doesn't *stick*. Your brain's thinking, "Better leave room for more important things, like which wild animals to avoid and whether naked snowboarding is a bad idea." So how *do* you trick your brain into thinking that your life depends on knowing Programming?

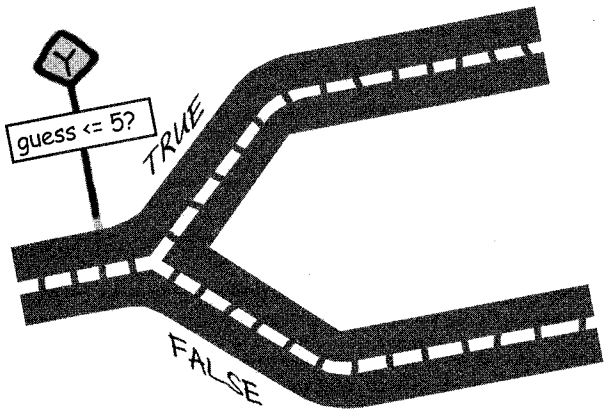
starting to code

1 Finding your way

Writing programs gives you the power to control your PC.

Almost everyone knows how to *use* a computer, but few people take the next step and learn how to *control* it. If you use other people's software, you will always be limited by what other people think you want to do. Write your own programs and the only limit will be your own imagination. Programming will make you more creative, it will make you think more precisely, and it will teach you to analyze and solve problems logically.

Do you want to be programmed or be the programmer?

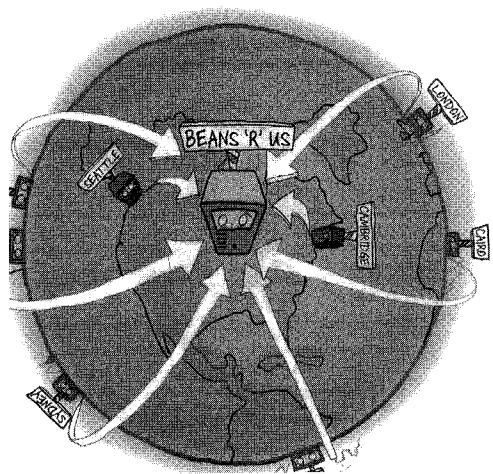


2 textual data

Every string has its place

Imagine trying to communicate without words.

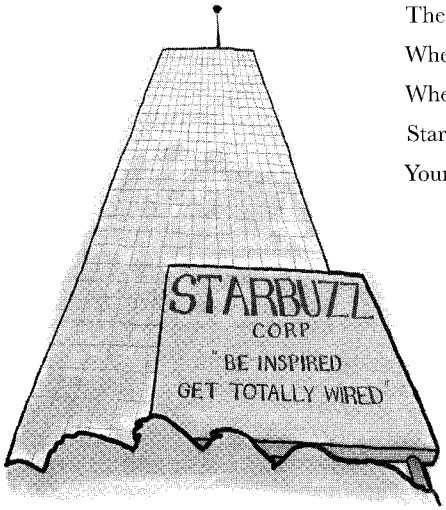
All programs process data, and one of the most important types of data is **text**. In this chapter, you'll work through the basics of **textual data**. You'll automatically **search** text and get back **exactly what you're looking for**. Along the way, you'll pick up key programming concepts such as **methods** and how you can use them to **bend your data to your will**. And finally, you'll instantly **power up your programs** with the help of **library code**.



3 functions

Let's get organized

As programs grow, the code often becomes more complex. And complex code can be hard to read, and even harder to maintain. One way of managing this complexity is to create **functions**. Functions are **snippets of code** that you use as needed from within your program. They allow you to **separate out common actions**, and this means that they make your code **easier to read** and **easier to maintain**. In this chapter, you'll discover how a little function knowledge can **make your coding life a whole lot easier**.

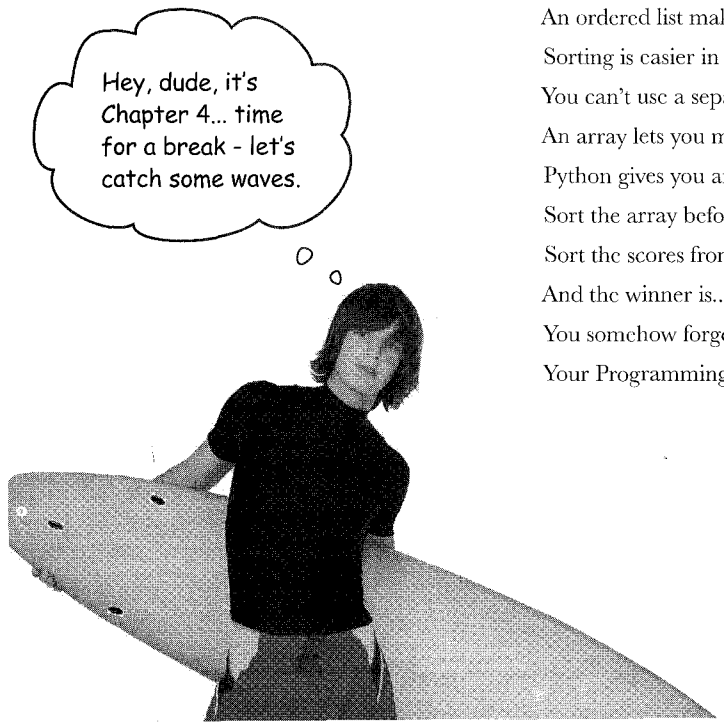


4 data files and arrays

Sort it out

As your programs develop, so do your data handling needs.

And when you have lots of data to work with, using an individual variable for each piece of data gets really old, really quickly. So programmers employ some rather awesome containers (known as **data structures**) to help them work with lots of data. More times than not, all that data comes from a file stored on a hard disk. So, how can you work with data in your files? Turns out it's a breeze.

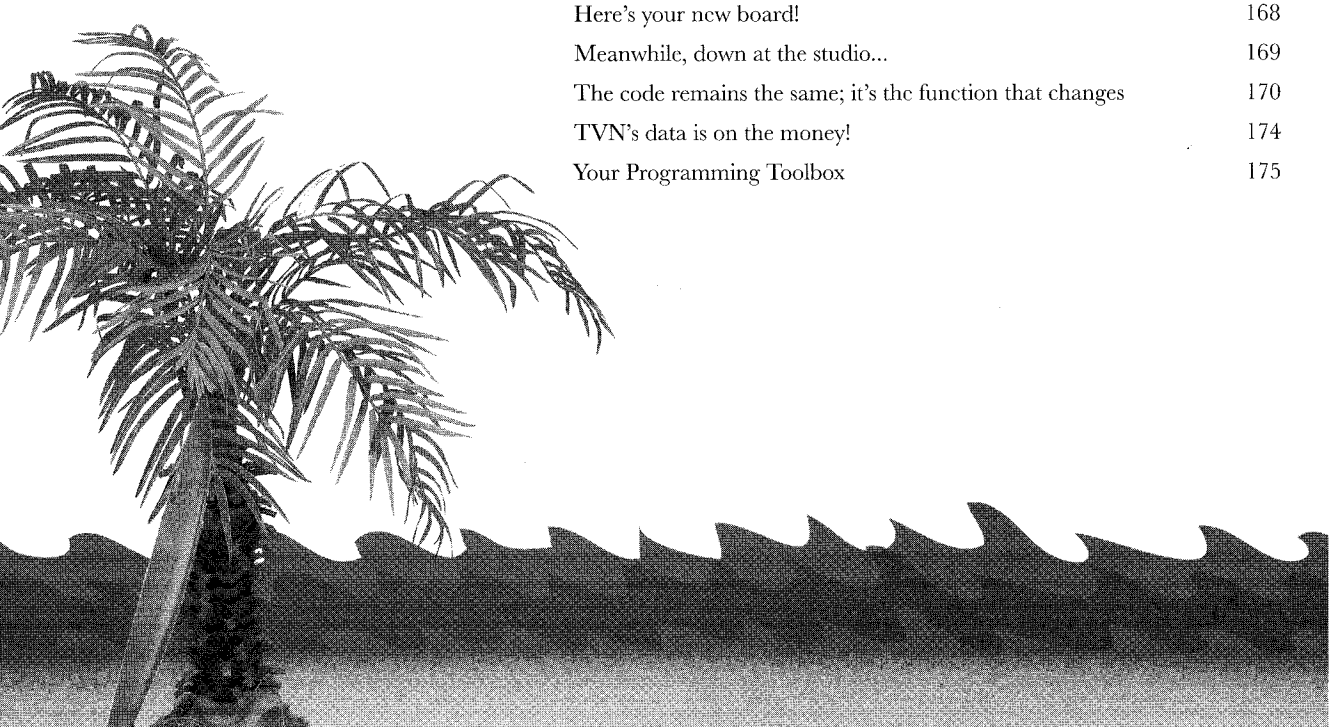


5 hashes and databases

Putting data in its place

Arrays aren't the only show in town when it comes to data.

Programming languages come with other data-arranging goodies too, and our chosen tool, Python, is no exception. In this chapter, you'll **associate** values with names using a data structure commonly called the **hash** (better known as *dictionary* to Python-folk). And when it comes to working with **stored data**, you'll read data from an *external database system* as well as from regular text-based files. All the world's awash with data, so turn the page and start applying your ever-expanding programming skills to some cool data-processing tasks.

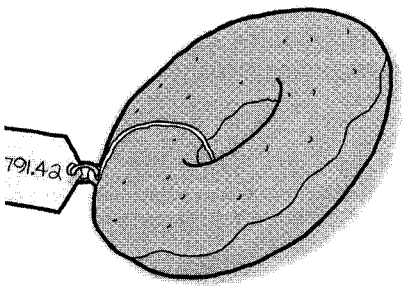


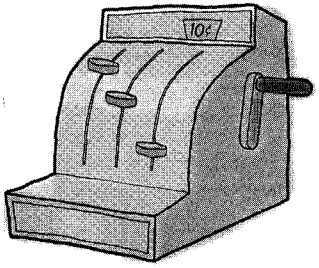
modular programming

6 Keeping things straight

The code that you write will make its way into many programs.

And, although **sharing** is good, you need to be *careful*. One programmer might take your code and use it in an **unexpected** way, while another might change it without even letting you know. You might want to use one function in all your programs and, over time, that function's code might **change** to suit your needs. Smart programmers take advantage of *modular programming techniques* to keep their workload manageable.



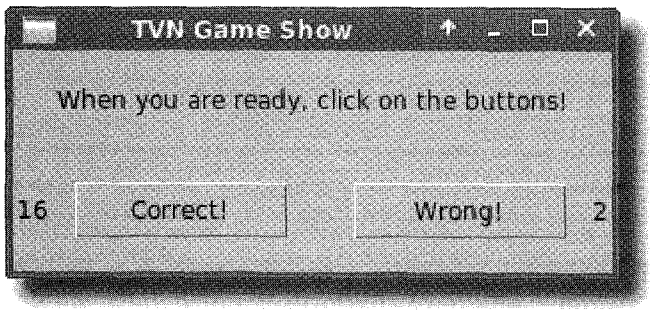


building a graphical user interface

7 Going all gooey

Your coding skills are great and getting better all the time.

It's just a shame your programs are not that *nice* to look at. Displaying prompts and messages on a text-based console is all well and good, but it's so 1970s, isn't it? Add some green text on a black background and your retro look will be complete. There has to be *a better way* to communicate with your users than the console, and there is: using a **graphical user interface** or **GUI** (pronounced "gooey"). Sounds cool, but complex, and it can be. But, don't fret; learning a trick or two will have your code all graphical in no time. Let's get all gooey (sorry, GUI) in this chapter.

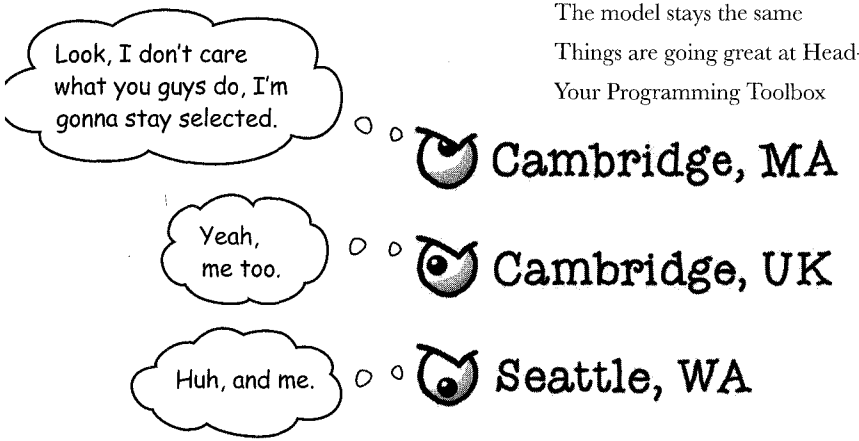


guis and data

8 Data entry widgets

GUIs don't just process events. They also handle data.

Almost all GUI applications need to read user data, and choosing the right widgets can change your interface from *data entry hell* to *user heaven*. Widgets can accept plain text, or just present a menu of options. There are lots of different widgets out there, which means there are lots of choices, too. And, of course, making the right choice can make all the difference. It's time to take your GUI program to the **next level**.



8½

exceptions and message boxes

Get the message?

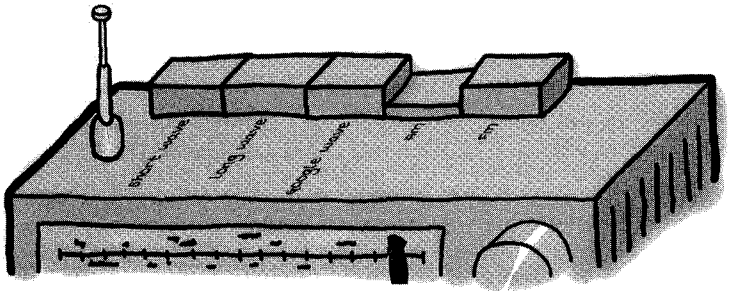
Sometimes things just go wrong. You just need to handle it.

There will always be things beyond your control. Networks will fail. Files will disappear. Smart coders learn how to deal with those kinds of **errors** and make their programs **recover** gracefully. The best software keeps the user informed about the bad things that happen and what should be done to recover. By learning how to use **exceptions** and **message boxes**, you can take your software to the next level of reliability and quality.

9 graphical interface elements

Selecting the right tool

It's easy to make your programs more effective for your users. And when it comes to GUI applications, there's a world of difference between a *working* interface and one that's both **useful** and **effective**. Selecting the right tool for the right job is a skill that comes with experience, and the best way to get that experience is to use the tools available to you. In this chapter, you'll continue to expand your GUI application building skills. There's a bunch of truly useful widgets waiting to be experienced. So, turn the page and let's get going.

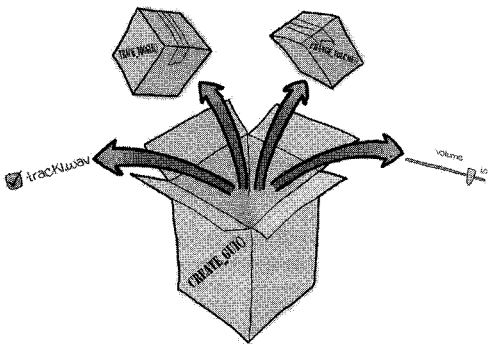


10 custom widgets and classes

With an object in mind

Requirements can be complex, but programs don't have to be.

By using object orientation, you can give your programs **great power** without writing lots of extra code. Keep reading, and you'll create **custom widgets** that do exactly what *you* want and give you the power to take **your programming skills to the next level**.



leftovers

The Top Ten Things (we didn't cover)

You've come a long way.

But learning how to program is an activity that never stops. The more you code, the more you'll need to **learn new ways to do certain things**. You'll need to master **new tools** and **new techniques**, too. There's just not enough room in this book to show you everything you might possibly need to know. So, here's our list of the top ten things we didn't cover that you might want to learn more about next.