Part I
Getting Started with Python

Visit www.dummies.com for great Dummies content online.

In this part . . .

- Discover what programming is all about and why you need Python to do it.
- Get your own copy of Python and install it on your system.
- Work with the interactive environment that Python provides.
- Create your first application using Python.
- Understand the benefits of adding comments to your application.

Chapter 1

Talking to Your Computer

In This Chapter

- Talking to your computer
- Creating programs to talk to your computer
- Understanding what a program does and why you want to create it
- Considering why you want to use Python as your programming language

Having a conversation with your computer might sound like the script of a science fiction movie. After all, the members of the *Enterprise* on *Star Trek* regularly talked with their computer. In fact, the computer often talked back. However, with the rise of Apple's Siri (`http://www.apple.com/ios/siri/`) and other interactive software, perhaps you really don't find a conversation so unbelievable.

Asking the computer for information is one thing, but providing it with instructions is quite another. This chapter considers why you want to instruct your computer about anything and what benefit you gain from it. You also discover the need for a special language when performing this kind of communication and why you want to use Python to accomplish it. However, the main thing to get out of this chapter is that programming is simply a kind of communication that is akin to other forms of communication you already have with your computer.

Understanding Why You Want to Talk to Your Computer

Talking to a machine may seem quite odd at first, but it's necessary because a computer can't read your mind — yet. Even if the computer did read your mind, it would still be communicating with you. Nothing can occur without an exchange of information between the machine and you. Activities such as

- Reading your e-mail
- Writing about your vacation
- Finding the greatest gift in the world

are all examples of communication that occurs between a computer and you. That the computer further communicates with other machines or people to address requests that you make simply extends the basic idea that communication is necessary to produce any result.

In most cases, the communication takes place in a manner that is nearly invisible to you unless you really think about it. For example, when you visit a chat room online, you might think that you're communicating with another person. However, you're communicating with your computer, your computer is communicating with the other person's computer through the chat room (whatever it consists of), and the other person's computer is communicating with that person. Figure 1-1 gives you an idea of what is actually taking place.

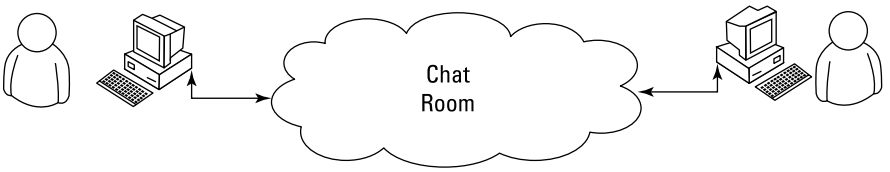

Figure 1-1: Communication with your computer may be invisible unless you really think about it.

Notice the cloud in the center of Figure 1-1. The cloud could contain anything, but you know that it at least contains other computers running other applications. These computers make it possible for your friend and you to chat. Now, think about how easy the whole process seems when you're using the chat application. Even though all these things are going on in the background, it seems as if you're simply chatting with your friend and the process itself is invisible.

Knowing that an Application is a Form of Communication

Computer communication occurs through the use of applications. You use one application to answer your e-mail, another to purchase goods, and still another to create a presentation. An *application* (sometimes called an *app*) provides the means to express human ideas to the computer in a manner the computer can understand and defines the tools needed to shape the data used for the communication in specific ways. Data used to express the content of a presentation is different from data used to purchase a present for your mother. The way you view, use, and understand the data is different for each task, so you must use different applications to interact with the data in a manner that both the computer and you can understand.

It's possible to obtain applications to meet just about any general need you can conceive of today. In fact, you probably have access to applications for which you haven't even thought about a purpose yet. Programmers have been busy creating millions of applications of all types for many years now, so it may be hard to understand what you can accomplish by creating some new method for talking with your computer through an application. The answer comes down to thinking about the data and how you want to interact with it. Some data simply isn't common enough to have attracted the attention of a programmer, or you may need the data in a format that no application currently supports, so you don't have any way to tell the computer about it unless you create a custom application to do it.

The following sections describe applications from the perspective of working with unique data in a manner that is special in some way. For example, you might have access to a video library database but no method to access it in a way that makes sense to you. The data is unique and your access needs are special, so you may want to create an application that addresses both the data and your needs.

Thinking about procedures you use daily

A *procedure* is simply a set of steps you follow to perform a task. For example, when making toast, you might use a procedure like this:

1. Get the bread and butter from the refrigerator.
2. Open the bread bag and take out two pieces of toast.
3. Remove the cover from the toaster.

4. Place each piece of bread in its own slot.
5. Push the toaster lever down to start toasting the bread.
6. Wait for the toasting process to complete.
7. Remove toast from the toaster.
8. Place toast on a plate.
9. Butter the toast.

Your procedure might vary from the one presented here, but it's unlikely that you'd butter the toast before placing it in the toaster. Of course, you do actually have to remove the bread from the wrapper before you toast it (placing the bread, wrapper and all, into the toaster would likely produce undesirable results). Most people never actually think about the procedure for making toast. However, you use a procedure like this one even though you don't think about it.

Computers can't perform tasks without a procedure. You must tell the computer which steps to perform, the order in which to perform them, and any exceptions to the rule that could cause failure. All this information (and more) appears within an application. In short, an application is simply a written procedure that you use to tell the computer what to do, when to do it, and how to do it. Because you've been using procedures all your life, all you really need to do is apply the knowledge you already possess to what a computer needs to know about specific tasks.

Writing procedures down

When I was in grade school, our teacher asked us to write a paper about making toast. After we turned in our papers, she brought in a toaster and some loaves of bread. Each paper was read and demonstrated. None of our procedures worked as expected, but they all produced humorous results. In my case, I forgot to tell the teacher to remove the bread from the wrapper, so she dutifully tried to stuff the piece of bread, wrapper and all, into the toaster. The lesson stuck with me. Writing about procedures can be quite hard because we know precisely want we want to do, but often we leave steps out — we assume that the other person also knows precisely what we want to do.

Many experiences in life revolve around procedures. Think about the checklist used by pilots before a plane takes off. Without a good procedure, the plane could crash. Learning to write a great procedure takes time, but it's doable. You may have to try several times before you get a procedure that

works completely, but eventually you can create one. Writing procedures down isn't really sufficient, though — you also need to test the procedure by using someone who isn't familiar with the task involved. When working with computers, the computer is your perfect test subject.

Seeing applications as being like any other procedure

A computer acts like the grade school teacher in my example in the previous section. When you write an application, you're writing a procedure that defines a series of steps that the computer should perform to accomplish whatever task you have in mind. If you leave out a step, the results won't be what you expected. The computer won't know what you mean or that you intended for it to perform certain tasks automatically. The only thing the computer knows is that you have provided it with a specific procedure and it needs to perform that procedure.

Understanding that computers take things literally

People eventually get used to the procedures you create. They automatically compensate for deficiencies in your procedure or make notes about things that you left out. In other words, people compensate for problems with the procedures that you write.

When you begin writing computer programs, you'll get frustrated because computers perform tasks precisely and read your instructions literally. For example, if you tell the computer that a certain value should equal 5, the computer will look for a value of exactly 5. A human might see 4.9 and know that the value is good enough, but a computer doesn't see things that way. It sees a value of 4.9 and decides that it doesn't equal 5 exactly. In short, computers are inflexible, unintuitive, and unimaginative. When you write a procedure for a computer, the computer will do precisely as you ask absolutely every time and never modify your procedure or decide that you really meant for it to do something else.

Defining What an Application Is

As previously mentioned, applications provide the means to define express human ideas in a manner that a computer can understand. To accomplish this goal, the application relies on one or more procedures that tell the computer how to perform the tasks related to the manipulation of data and its presentation. What you see onscreen is the text from your word processor, but to see that information, the computer requires procedures for retrieving the data from disk, putting it into a form you can understand, and then presenting it to you. The following sections define the specifics of an application in more detail.

Understanding that computers use a special language

Human language is complex and difficult to understand. Even applications such as Siri have serious limits in understanding what you're saying. Over the years, computers have gained the capability to input human speech as data and to understand certain spoken words as commands, but computers still don't quite understand human speech to any significant degree. The difficulty of human speech is exemplified in the way lawyers work. When you read legalese, it appears as a gibberish of sorts. However, the goal is to state ideas and concepts in a way that isn't open to interpretation. Lawyers seldom succeed in meeting their objective precisely because human speech is imprecise.

Given what you know from previous sections of this chapter, computers could never rely on human speech to understand the procedures you write. Computers always take things literally, so you'd end up with completely unpredictable results if you were to use human language to write applications. That's why humans use special languages, called *programming languages,* to communicate with computers. These special languages make it possible to write procedures that are both specific and completely understandable by both humans and computers.

Computers don't actually speak any language. They use binary codes to flip switches internally and to perform math calculations. Computers don't even understand letters — they understand only numbers. A special application turns the computer-specific language you use to write a procedure into binary codes. For the purposes of this book, you really don't need to worry too much about the low-level specifics of how computers work at the binary level. However, it's interesting to know that computers speak math and numbers, not really a language at all.

Helping humans speak to the computer

It's important to keep the purpose of an application in mind as you write it. An application is there to help humans speak to the computer in a certain way. Every application works with some type of data that is input, stored, manipulated, and output so that the humans using the application obtain a desired result. Whether the application is a game or a spreadsheet, the basic idea is the same. Computers work with data provided by humans to obtain a desired result.

When you create an application, you're providing a new method for humans to speak to the computer. The new approach you create will make it possible for other humans to view data in new ways. The communication between human and computer should be easy enough that the application actually disappears from view. Think about the kinds of applications you've used in the past. The best applications are the ones that let you focus on whatever data you're interacting with. For example, a game application is considered immersive only if you can focus on the planet you're trying to save or the ship you're trying to fly, rather than the application that lets you do these things.

One of the best ways to start thinking about how you want to create an application is to look at the way other people create applications. Writing down what you like and dislike about other applications is a useful way to start discovering how you want your applications to look and work. Here are some questions you can ask yourself as you work with the applications:

- What do I find distracting about the application?
- Which features were easy to use?
- Which features were hard to use?
- How did the application make it easy to interact with my data?
- How would I make the data easier to work with?
- What do I hope to achieve with my application that this application doesn't provide?

Professional developers ask many other questions as part of creating an application, but these are good starter questions because they begin to help you think about applications as a means to help humans speak with computers. If you've ever found yourself frustrated by an application you used, you already know how other people will feel if you don't ask the appropriate questions when you create your application. Communication is the most important element of any application you create.

You can also start to think about the ways in which you work. Start writing procedures for the things you do. It's a good idea to take the process one step at a time and write everything you can think of about that step. When you get finished, ask someone else to try your procedure to see how it actually works. You might be surprised to learn that even with a lot of effort, you can easily forget to include steps.

The world's worst application usually begins with a programmer who doesn't know what the application is supposed to do, why it's special, what need it addresses, or whom it is for. When you decide to create an application, make sure that you know why you're creating it and what you hope to achieve. Just having a plan in place really helps make programming fun. You can work on your new application and see your goals accomplished one at a time until you have a completed application to use and show off to your friends (all of whom will think you're really cool for creating it).

Understanding Why Python is So Cool

Many programming languages are available today. In fact, a student can spend an entire semester in college studying computer languages and still not hear about them all. (I did just that during my college days.) You'd think that programmers would be happy with all these programming languages and just choose one to talk to the computer, but they keep inventing more.

Programmers keep creating new languages for good reason. Each language has something special to offer — something it does exceptionally well. In addition, as computer technology evolves, so do the programming languages in order to keep up. Because creating an application is all about efficient communication, many programmers know multiple programming languages so that they can choose just the right language for a particular task. One language might work better to obtain data from a database, and another might create user interface elements especially well.

As with every other programming language, Python does some things exceptionally well, and you need to know what they are before you begin using it. You might be amazed by the really cool things you can do with Python. Knowing a programming language's strengths and weaknesses helps you use it better as well as avoid frustration by not using the language for things it doesn't do well. The following sections help you make these sorts of decisions about Python.

Unearthing the reasons for using Python

Most programming languages are created with specific goals in mind. These goals help define the language characteristics and determine what you can do with the language. There really isn't any way to create a programming language that does everything because people have competing goals and needs when creating applications. When it comes to Python, the main objective was to create a programming language that would make programmers efficient and productive. With that in mind, here are the reasons that you want to use Python when creating an application:

- **Less application development time:** Python code is usually 2–10 times shorter than comparable code written in languages like C/C++ and Java, which means that you spend less time writing your application and more time using it.
- **Ease of reading:** A programming language is like any other language — you need to be able to read it to understand what it does. Python code tends to be easier to read than the code written in other languages, which means you spend less time interpreting it and more time making essential changes.
- **Reduced learning time:** The creators of Python wanted to make a programming language with fewer odd rules that make the language hard to learn. After all, programmers want to create applications, not learn obscure and difficult languages.

It's important to realize that, although Python is a popular language, it's not the most popular language out there. In fact, it currently ranks eighth on sites such as TIOBE (`http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`), an organization that tracks usage statistics (among other things). If you're looking for a language solely for the purpose of obtaining a job, Python is a good choice, but C/C++, Java, C#, or Visual Basic would be better choices. Make sure you choose a language you like and one that will address your application development needs, but also choose on the basis of what you intend to accomplish. Python was the language of the year in both 2007 and 2010 and has ranked as high as the fourth most popular language in February 2011. So, really, it's a good choice if you're looking for a job, but not necessarily the best choice. However, it may surprise you to know that many colleges now use Python to teach coding, and it has become the most popular language in that venue. Check out my blog post at `http://blog.johnmuellerbooks.com/2014/07/14/python-as-a-learning-tool` for details.

Deciding how you can personally benefit from Python

Ultimately, you can use any programming language to write any sort of application you want. If you use the wrong programming language for the job, the process will be slow, error prone, bug ridden, and you'll absolutely hate it — but you can get the job done. Of course, most of us would rather avoid horribly painful experiences, so it's important to know what sorts of applications people typically use Python to create. Here's a list of the most common uses for Python (although people do use it for other purposes):

- **Creating rough application examples:** Developers often need to create a *prototype,* a rough example of an application, before getting the resources to create the actual application. Python emphasizes productivity, so you can use it to create prototypes of an application quickly.
- **Scripting browser-based applications:** Even though JavaScript is probably the most popular language used for browser-based application scripting, Python is a close second. Python offers functionality that JavaScript doesn't provide (see the comparison at `https://blog.glyphobet.net/essay/2557` for details) and its high efficiency makes it possible to create browser-based applications faster (a real plus in today's fast-paced world).
- **Designing mathematic, scientific, and engineering applications:** Interestingly enough, Python provides access to some really cool libraries that make it easier to create math, scientific, and engineering applications. The two most popular libraries are NumPy (`http://www.numpy.org/`) and SciPy (`http://www.scipy.org/`). These libraries greatly reduce the time you spend writing specialized code to perform common math, scientific, and engineering tasks.
- **Working with XML:** The eXtensible Markup Language (XML) is the basis of most data storage needs on the Internet and many desktop applications today. Unlike most languages, where XML is just sort of bolted on, Python makes it a first-class citizen. If you need to work with a Web service, the main method for exchanging information on the Internet (or any other XML-intensive application), Python is a great choice.
- **Interacting with databases:** Business relies heavily on databases. Python isn't quite a query language, like the Structure Query Language (SQL) or Language INtegrated Query (LINQ), but it does do a great job of interacting with databases. It makes creating connections and manipulating data relatively painless.

- **Developing user interfaces:** Python isn't like some languages like C# where you have a built-in designer and can drag and drop items from a toolbox onto the user interface. However, it does have an extensive array of graphical user interface (GUI) frameworks — extensions that make graphics a lot easier to create (see `https://wiki.python.org/moin/GuiProgramming` for details). Some of these frameworks do come with designers that make the user interface creation process easier. The point is that Python isn't devoted to just one method of creating a user interface — you can use the method that best suits your needs.

Discovering which organizations use Python

Python really is quite good at the tasks that it was designed to perform. In fact, that's why a lot of large organizations use Python to perform at least some application-creation (development) tasks. You want a programming language that has good support from these large organizations because these organizations tend to spend money to make the language better. Here's a list of the large organizations that use Python the most:

- Alice Educational Software – Carnegie Mellon University (`http://www.cmu.edu/corporate/news/2007/features/alice.shtml`): Educational applications
- Fermilab (`https://www.fnal.gov/`): Scientific applications
- Go.com (`http://go.com/`): Browser-based applications
- Google (`https://www.google.com/`): Search engine
- Industrial Light & Magic (`http://www.ilm.com/`): Just about every programming need
- Lawrence Livermore National Library (`https://www.llnl.gov/`): Scientific applications
- National Space and Aeronautics Administration (NASA) (`http://www.nasa.gov/`): Scientific applications
- New York Stock Exchange (`https://nyse.nyx.com/`): Browser-based applications
- ObjectDomain (`http://case-tools.org/tools/objectdomain.html`): Computer Aided Software Engineering (CASE) tools
- Redhat (`http://www.redhat.com/`): Linux installation tools

- Yahoo! (`https://www.yahoo.com/`): Parts of Yahoo! mail
- YouTube (`http://www.youtube.com/`): Graphics engine
- Zope – Digital Creations (`http://www.zope.com/`): Publishing application

These are just a few of the many organizations that use Python extensively. You can find a more complete list of organizations at `http://www.python.org/about/success/`. The number of success stories has become so large that even this list probably isn't complete and the people supporting it have had to create categories to better organize it.

Finding useful Python applications

You might have an application written in Python sitting on your machine right now and not even know it. Python is used in a vast array of applications on the market today. The applications range from utilities that run at the console to full-fledged CAD/CAM suites. Some applications run on mobile devices, while others run on the large services employed by enterprises. In short, there is no limit to what you can do with Python, but it really does help to see what others have done. You can find a number of places online that list applications written in Python, but the best place to look is `https://wiki.python.org/moin/Applications`.

As a Python programmer, you'll also want to know that Python development tools are available to make your life easier. A *development tool* provides some level of automation in writing the procedures needed to tell the computer what to do. Having more development tools means that you have to perform less work in order to obtain a working application. Developers love to share their lists of favorite tools, but you can find a great list of tools broken into categories at `http://www.python.org/about/apps/`.

Of course, this chapter describes a number of tools as well, such as NumPy and SciPy (two scientific libraries). The remainder of the book lists a few other tools; make sure that you copy down your favorite tools for later.

Comparing Python to other languages

Comparing one language to another is somewhat dangerous because the selection of a language is just as much a matter of taste and personal preference as it is any sort of quantifiable scientific fact. So before I'm attacked by the rabid protectors of the languages that follow, it's important to realize

that I also use a number of languages and find at least some level of overlap among them all. There is no best language in the world, simply the language that works best for a particular application. With this idea in mind, the following sections provide an overview comparison of Python to other languages. (You can find comparisons to other languages at `https://wiki.python.org/moin/LanguageComparisons`.)

C#

A lot of people claim that Microsoft simply copied Java to create C#. That said, C# does have some advantages (and disadvantages) when compared to Java. The main (undisputed) intent behind C# is to create a better kind of C/C++ language — one that is easier to learn and use. However, we're here to talk about C# and Python. When compared to C#, Python has these advantages:

- Significantly easier to learn
- Smaller (more concise) code
- Supported fully as open source
- Better multiplatform support
- Easily allows use of multiple development environments
- Easier to extend using Java and C/C++
- Enhanced scientific and engineering support

Java

For years, programmers looked for a language that they could use to write an application just once and have it run anywhere. Java is designed to work well on any platform. It relies on some tricks that you'll discover later in the book to accomplish this magic. For now, all you really need to know is that Java was so successful at running well everywhere that other languages have sought to emulate it (with varying levels of success). Even so, Python has some important advantages over Java, as shown in the following list:

- Significantly easier to learn
- Smaller (more concise) code
- Enhanced variables (storage boxes in computer memory) that can hold different kinds of data based on the application's needs while running (dynamic typing)
- Faster development times

Perl

PERL was originally an acronym for Practical Extraction and Report Language. Today, people simply call it Perl and let it go at that. However, Perl still shows its roots in that it excels at obtaining data from a database and presenting it in report format. Of course, Perl has been extended to do a lot more than that — you can use it to write all sorts of applications. (I've even used it for a Web service application.) In a comparison with Python, you'll find that Python has these advantages over Perl:

- Simpler to learn
- Easier to read
- Enhanced protection for data
- Better Java integration
- Fewer platform-specific biases